ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники
Кафедра систем автоматизации управления

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему
Разработка модуля проектирования виртуальных лабораторных стендов

Пояснительная записка

Киров 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники
Кафедра систем автоматизации управления

Допускаю к защите

Заведующий кафедрой САУ _____ /Ланских Ю.В./

(подпись) (Ф.И.О)

Разработка модуля проектирования виртуальных лабораторных стендов

Пояснительная записка выпускной квалификационной работы

ТПЖА.090302.036 ПЗ

Разработал студент группы ИТб-4301-01-00	(подпись)	/ Дождиков И.С./ (Ф.И.О)	25 июня 2020
Руководитель, к.т.н., доцент	(подпись)	/Ланских Ю.В./ (Ф.И.О)	25 июня 2020
Консультанты:			
по организационно-экономическому разделу, к.т.н., доцент	(подпись)	/ Ланских Ю.В./ (Ф.И.О)	25 июня 2020
Нормоконтролер, к.т.н., доцент	(подпись)	/ Ланских Ю.В./ (Ф.И.О)	25 июня 2020

Киров 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра ___САУ_____

УТВЕРЖДАЮ
Зав. кафедрой _Ланских Ю.В.__
«___» _____ 20 ___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студенту ___группы ИТб-4301-01-00 Дождикову Игорю Сергеевичу, номер зачетной книжки Д10-ФАВТ-2016-36_____

_____

Тема: __Разработка модуля проектирования виртуальных лабораторных стендов_____

_____

(Утверждена приказом по университету от __.___.____ № _____)

1. Исходные данные к ВКР _язык программирования C# 7.3, платформа .NET Framework 4.5.2, ЭВМ офисного назначения с ОС Microsoft Windows 10, Microsoft Office 2010, Git 2.23.0, Microsoft Visual Studio Community 2019 16.3.0; системные требования к рабочему месту пользователя: ЭВМ офисного назначения с установленной платформой .NET Framework версии 4.5.2 или выше, не менее 100 Мбайт объема оперативной памяти, не менее 50 Мбайт свободного места на жестком диске, процессор с частотой не менее 1 ГГц_____

2. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов) __обзор современного состояния предметной области виртуальных лабораторных стендов, проектирование модуля проектирования виртуальных лабораторных стендов, разработка модуля проектирования САПР LabCAD, технико-экономическое обоснование разработки модуля проектирования_____

_____

_____

3. Задание по организационной и экономической части проекта _расчет затрат на создание программного обеспечения, цены продукта; расчет выручки и прибыли от реализации программного продукта; расчёт затрат, связанных с покупкой, внедрением и использованием программного обеспечения_____

4. Задание по экологии и технике безопасности _____

_____

_____

_____

5. Прочие разделы _____

_____

6. Перечень графического материала (с точным указанием обязательных чертежей)

_____

_____

_____

7. Руководитель и консультант по ВКР (с указанием фамилии, имени, отчества, места работы и должности)

_____

_____

а) руководитель проекта _Ланских Юрий Владимирович, ФГБОУ ВО «Вятский государственный университет» (ВятГУ), доцент кафедры Систем автоматизации управления _____

б) консультант по организационно-экономической части _Ланских Юрий Владимирович, ФГБОУ ВО «Вятский государственный университет» (ВятГУ), доцент кафедры Систем автоматизации управления_____

в) консультант по технике безопасности _____

_____

г) консультант по _____

_____

д) консультант по _____

_____

8. Дата выдачи задания _20 апреля 2020 года_____

Руководитель _____ (подпись)

Студент _____ (подпись)

КАЛЕНДАРНЫЙ ГРАФИК

работы над ВКР на весь период проектирования с указанием объема выполнения и трудоемкости отдельных этапов по месяцам

1._Обзор предметной области и формирование требований – 30.04.2020_____

2._Проектирование модуля проектирования виртуальных лабораторных стендов – 08.05.2020_____

3._Разработка модуля проектирования – 15.05.2020_____

4._Технико-экономическое обоснование разработки модуля – 22.05.2020_____

5._Оформление расчетно-пояснительной записки – 29.05.2020_____

6. Даты: предзащиты ВКР _01.06.2020_ защиты _09.07.2020__

Руководитель _____ (подпись)

Студент _____ (подпись)

Реферат

Дождиков И.С. Разработка модуля проектирования виртуальных лабораторных стендов: ТПЖА.090302.036 ПЗ: Выпускная квалификационная работа / ВятГУ, каф. САУ; рук. Ю.В. Ланских. – Киров, 2020. ПЗ 150 с., 86 рис., 11 табл., 8 источников, 13 прил.; програм. докум. 10 л.

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА, СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ, МОДЕЛЬ, ВИРТУАЛЬНЫЙ ЛАБОРАТОРНЫЙ СТЕНД, МОДУЛЬ, ОБЪЕКТ МОДЕЛИ, ПЕРЕМЕННАЯ ОБЪЕКТА, ПОВЕДЕНИЕ ОБЪЕКТА, ВИЗУАЛЬНЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ.

Объект исследования и разработки – модуль проектирования виртуальных лабораторных стендов.

Цель работы – уменьшение использования временных и иных видов ресурсов, связанных с разработкой виртуальных лабораторных стендов, за счет автоматизации данного процесса модулем проектирования виртуальных лабораторных стендов.

Изучена предметная область и определена концепция модели для виртуальных лабораторных стендов, разработаны диаграммы IDEF0, IDEF3, DFD модуля проектирования виртуальных лабораторных стендов, описаны модели данных, спроектирован синтаксис нижнего уровня визуального языка программирования LabScript, произведено моделирование модуля с использованием языка UML. На основе результатов проектирования разработана программная реализация модуля проектирования виртуальных лабораторных стендов и демонстрационный стенд. Произведено технико-экономическое обоснование разработки модуля.

№ строки	Формат	Обозначение	Наименование	Кол-во листов	№ экз.	Примеч
1			Документация общая			
2			Вновь разработанная			
3						
4		ТПЖА.090302.036 ПЗ	Пояснительная записка	150		
5			в том числе			
6	А4		страниц формата А4	110		
7	А3		страниц формата А3	40		
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА.090302.036 ДДП		
Разраб.		Дождиков И.С.			Разработка модуля проектирования виртуальных лабораторных стендов	Литер	Лист 1 / Листов 1
Пров.		Ланских Ю.В.					
Т.контр		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Содержание

Введение

Важное место в жизни современного человека занимает использование различного рода аппаратно-программных средств. Программное обеспечение участвует в решении широкого круга задач: от простого набора текста до управления объектами ядерной энергетики. Одним из направлений использования программ является автоматизация различных процессов с целью минимизации затрачиваемых ресурсов.

В данной работе пойдет речь об автоматизации проектирования виртуальных лабораторных стендов через создание модуля проектирования виртуальных лабораторных стендов.

Вся работа разбита на несколько структурированных разделов с целью подачи информации в удобном для читателя виде.

Первый раздел, «Обзор современного состояния предметной области виртуальных лабораторных стендов», содержит комплексные сведения о предметной области и объекте разработки для осуществления процессов проектирования и кодирования.

Второй раздел, «Проектирование модуля проектирования виртуальных лабораторных стендов», посвящен идейным основам модуля и вопросам проектирования.

Третий раздел, «Разработка модуля проектирования САПР LabCAD», описывает результаты кодирования модуля и разработку демонстрационного стенда.

Четвертый раздел, «Технико-экономическое обоснование разработки модуля проектирования», посвящен экономическим вопросам разработки модуля.

Каждый раздел сопровождается графическим материалом для лучшего понимания соответствующих аспектов разработки модуля проектирования виртуальных лабораторных стендов.

1 Обзор современного состояния предметной области виртуальных лабораторных стендов

Для разработки модуля проектирования необходимо рассмотреть соответствующую предметную область виртуальных лабораторных стендов, сформировать правильное направление разработки конечного продукта, определить требования к разрабатываемому модулю и, конечно, рассмотреть уже имеющиеся аналоги для подчеркивания необходимости реализации данной программы.

1.1 Описание предметной области

В настоящее время с учетом наличия у образовательных учреждений комплекса ЭВМ особое значение в образовательном процессе имеет использование различного рода программных решений, ориентированных на повышение качества получения знаний, в том числе за счет создания виртуальных моделей реальных процессов и явлений.

Последние можно назвать следующим понятием – виртуальные лабораторные стенды.

Виртуальный лабораторный стенд (ВЛС, стенд) – это многофункциональный программно-методический комплекс, моделирующий действие изучаемых явлений, приборов, механизмов и технических средств [1].

Преимущества виртуальных лабораторных стендов перед реальными состоят в следующем:

- отсутствие необходимости приобретения дорогостоящего оборудования и реактивов;
- возможность моделирования процессов, протекание которых принципиально невозможно в лабораторных условиях;
- безопасность проведения опытов в виртуальной среде;
- возможность быстрого проведения серии опытов с различными значениями входных параметров;
- возможность использования виртуальной лаборатории в дистанционном обучении.

Виртуальные лабораторные стенды в основе своей содержат совокупность математических моделей, описывающих в определенной степени точности протекающие в реальном мире процессы и явления. Но эти модели не взаимодействуют с пользователем сами по себе, так как, по сути, представляют собой лишь математические зависимости между входными и выходными моделируемыми величинами, а обслуживаются

специализированной программой. Обслуживающая программа может выступать как в роли отдельного исполняемого файла или веб-приложения со встроенными в них математическими моделями, так и программой-интерпретатором, в которую подгружаются модели, закодированные в специальном формате, доступном для чтения и исполнения данной программой.

Понятие «встроенный» здесь имеет значение внедрения формализованного описания математической модели непосредственно в программный код обслуживающей программы.

Программа уже за счет использования различного вида пользовательских интерфейсов осуществляет взаимодействие с пользователем, принимая от него входные данные и выводя рассчитанные на основе математических моделей выходные значения.

Сам процесс разработки ВЛС может занимать продолжительное время и требует от разработчика ВЛС не только глубокого понимания предметной области, но и наличия достаточно высоких навыков программирования для программной реализации виртуального лабораторного стенда, либо наличия значительных финансовых ресурсов для включения в процесс разработки специализированной фирмы по разработке программного обеспечения. Данную проблему позволяют решить системы автоматизированного проектирования виртуальных лабораторных стендов.

Говоря о понятиях «система автоматизированного проектирования» и «автоматизированное проектирование», необходимо понимать, что вкладывается вообще в понятие «проектирование».

Проектирование – процесс составления описания, необходимого для создания в заданных условиях еще несуществующего объекта, на основе первичного описания этого объекта и (или) алгоритма его функционирования [2, с. 5].

Проектирование представляет собой достаточно сложный вид человеческой деятельности, который невозможен без наличия соответствующих глубоких знаний и способности к творческому поиску, который, однако, включает в себя выполнение некоторой рутинной работы.

Данный процесс можно оптимизировать по использованию различного рода ресурсов за счет его автоматизации при помощи ЭВМ. Это достигается путем рационального распределения функций между проектировщиком и ЭВМ, причем на проектировщика возлагаются, как правило, задачи творческого характера, тогда как на ЭВМ – задачи, допускающие решение с помощью некоторого определенного алгоритма.

Выполнением формализованных задач процесса проектирования в настоящее время занимаются системы автоматизированного проектирования (САПР). САПР – это системы, реализующие автоматизированное проектирование [3, с. 12-13]. Классификация САПР возможна по множеству критериев, причем, в большинстве случаев конкретная классификация носит условный характер.

Таким образом, САПР для проектирования виртуальных лабораторных стендов представляет собой автоматизированную систему, позволяющую пользователю разработать виртуальный лабораторный стенд для моделирования объектов и процессов реального мира в соответствии с требованиями пользователя на проблемно-ориентированном уровне с описаниями алгоритмов на визуальном языке программирования при наличии базовых навыков программирования.

Под визуальным языком программирования в настоящей работе понимается совокупность графических или текстографических объектов, манипулируя которыми пользователь задает логику работы модели.

Следует заметить, что понятия «модель» и «виртуальный лабораторный стенд» в рамках данной работы являются синонимами.

1.2 Определение цели и задач проекта

Система автоматизированного проектирования представляет собой совокупность взаимосвязанных подсистем или модулей, каждый из которых предназначен для решения конкретного списка подзадач в рамках единой задачи, которую должна решать САПР. Целью данной работы является уменьшение использования временных и иных видов ресурсов, связанных с разработкой виртуальных лабораторных стендов, за счет автоматизации данного процесса модулем проектирования ВЛС. Данный модуль должен предоставить пользователю необходимый интерфейс для обеспечения процесса проектирования ВЛС, а также средства для получения описания моделируемой системы и для конвертации описания моделируемой системы в файл. Задачи, которые необходимо выполнить для достижения поставленной цели:

1) рассмотреть аналоги для формирования особенностей разрабатываемого модуля;
2) спроектировать модуль проектирования ВЛС:
   а) определить концепцию модели, разрабатываемой в модуле проектирования ВЛС;
   б) разработать функциональные модели модуля в нотациях IDEF0, IDEF3, DFD;

в) определить структуры данных, с которыми будет взаимодействовать модуль проектирования ВЛС для сохранения и загрузки моделей;
г) определить грамматику визуального языка программирования LabScript;
д) разработать объектно-ориентированные модели организации архитектуры модуля с использованием языка UML.

3) разработать программную реализацию модуля проектирования ВЛС:
а) выбрать язык программирования;
б) реализовать модуль с использованием выбранного языка программирования;
в) разработать демонстрационный пример.

4) произвести технико-экономическое обоснование разработки модуля проектирования:
а) рассчитать затраты на создание модуля и на его цену;
б) рассчитать выручку и прибыль от реализации;
в) рассчитать затраты, связанные с покупкой, внедрением и использованием модуля проектирования ВЛС.

1.3 Формирование требований

Модуль проектирования виртуальных лабораторных стендов (ВЛС) является частью системы автоматизированного проектирования LabCAD, которая состоит из следующих модулей: модуль обслуживания, модуль интерпретатора, модуль компилятора. Модуль обслуживания предназначен для выполнения служебных операций с системой, например, поиск актуальных обновлений модулей и их загрузка. Модуль интерпретатора предназначен для выполнения процесса симуляции, при котором обучающийся может взаимодействовать со ВЛС и получать результаты реакции ВЛС на действия обучающегося. Модуль компилятора предназначен для создания из файла стенда исполняемого автономного приложения.

Для модуля определена одна группа пользователей – преподаватели, заинтересованные в уменьшении временных и иных видов затрат на разработку ВЛС при наличии базовых навыков разработки программных систем. Соответственно, с точки зрения преподавателей модуль проектирования ВЛС должен обеспечивать автоматизацию процесса создания ВЛС в части определения структуры стенда и использования расширений, содержащих необходимый набор функций для реализации ВЛС в рамках конкретной предметной области.

К используемым модулем данным предъявляются следующие требования: они должны поддерживаться языком программирования C# 7.3,

поддерживать сериализацию и десериализацию для обеспечения процесса сохранения и загрузки ВЛС.

Далее следует описать функционал модуля проектирования ВЛС.

Преподаватель при взаимодействии с модулем может работать с моделями: создавать новые модели и загружать уже существующие. Каждая модель состоит из двух частей: метаданных модели, включающих в себя краткие сведения о модели (название модели, версия модели, автор модели, краткое описание модели) и данные о полях проектирования, и совокупности взаимосвязанных между собой объектов модели.

Связь между объектами модели подразумевает то, что они предоставляют некоторые свои переменные и доступ к публичным функциям связанным с ними объектам.

Преподаватель может работать с расширениями, то есть группами объектов модели конкретной предметной области, которые он будет использовать в своем проекте: подключать и отключать их.

Преподаватель может взаимодействовать с полями проектирования для размещения объектов модели в пределах данных полей, установления связей между объектами (доступно только для проектирования логики), изменения размеров объектов. Всего имеются два вида полей проектирования: поле проектирования логики и поле проектирования интерфейса.

Поле проектирования логики предназначено для организации внутренней структуры ВЛС и организации взаимодействия объектов модели между собой.

Поле проектирования интерфейса предназначено для формирования графического интерфейса ВЛС, предоставляемого обучающемуся.

Преподаватель может менять как переменные объектов (добавлять новые, удалять и модифицировать имеющиеся, за исключением программных переменных, которые предоставляют разработчики расширений и которые необходимы для правильного функционирования объектов модели), так и добавлять поведение объектам.

Поведение объекта представляет собой иерархическую древовидную структуру простых конструкций (присваивание значения переменной объекта модели, вызов функции объекта) и управляющих конструкций (циклов, условий), вложенных в программы, с помощью которых пользователь может реализовать логику работы объекта в рамках проектируемого ВЛС, что является дополнением к базовому функционалу объекта, заложенного в него разработчиком расширения.

Преподаватель в конце процесса проектирования может сохранить модель в виде файла в удобном для него месте.

С учетом выбранной платформы разработки .NET Framework работа данного модуля возможна только на ЭВМ под управлением ОС Windows и с наличием платформы .NET Framework. В настоящей работе будет использована версия 4.5.2 указанной платформы разработки.

1.4 Обзор существующих аналогов программной реализации

При поиске имеющихся на рынке программных аналогов, предоставляющих средства для создания виртуальных лабораторных стендов, были найдены следующие программные системы: Simulink, LabVIEW, JMCAD, VisSim. Последние две системы, согласно датам выхода последних версий и активности на официальных сайтах, перестали развиваться, поэтому могут не рассматриваться как конкуренты в данной области. Однако программа JMCAD разработана украинским разработчиком, что делает ее наиболее близкой к программам отечественного производства, поэтому она также будет рассмотрена наравне с первыми двумя из найденных. Сравнение данных систем можно рассмотреть по следующим критериям: стоимость, разграничение интерфейсов пользователя и разработчика, возможность расширения стандартной библиотеки, целевые платформы, использование визуальных языков программирования, ориентация на создание обучающих программных систем.

1.4.1 Графическая среда Simulink

Simulink – это графическая среда имитационного моделирования, позволяющая при помощи блок-диаграмм в виде направленных графов, строить динамические модели, включая дискретные, непрерывные и гибридные, нелинейные и разрывные системы.

Она позволяет использовать уже готовые библиотеки блоков для моделирования электросиловых, механических и гидравлических систем, а также применять развитый модельно-ориентированный подход при разработке систем управления, средств цифровой связи и устройств реального времени.

Средства моделирования Simulink основываются на программных средствах MATLAB, но позволяют обойтись без использования в явном виде языка MATLAB и создавать модели из стандартных блоков в графическом виде.

Экранная форма интерфейса Simulink представлена на рисунке 1.

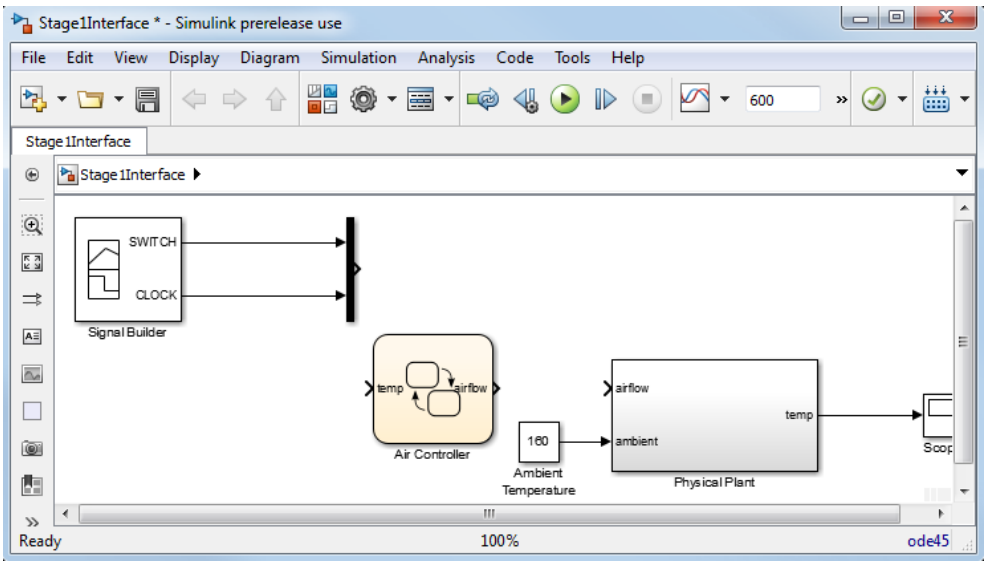


Рисунок 1 – Экранная форма интерфейса Simulink

Графическая среда Simulink имеет следующие достоинства:

- настройка пользовательского интерфейса для модели;
- возможность расширения стандартной библиотеки элементов;
- поддержка основных платформ (Microsoft Windows, Linux, macOS);
- наличие визуального языка программирования, представленного в виде соответствующих элементов в стандартной библиотеке, а также расширениях;
- наличие необходимых средств для реализации обучающих программных систем.

Недостатки у графической среды следующие:

- высокая стоимость (версия для академического использования: 275 долларов в год; 550 долларов - бессрочно);
- относительная сложность организации взаимодействия пользовательского интерфейса с моделью;
- ограниченные возможности визуального языка программирования.

Последней версией графической среды является Simulink R2020a.

1.4.2 Программное обеспечение LabVIEW

LabVIEW – это среда разработки и платформа для выполнения программ, созданных на графическом языке программирования «G» фирмы National Instruments (США).

LabVIEW используется в системах сбора и обработки данных, а также для управления техническими объектами и технологическими процессами. Идеологически LabVIEW очень близка к SCADA-системам, но в отличие от

них в большей степени ориентирована на решение задач не столько в области АСУ ТП, сколько в области АСНИ.

Графический язык программирования «G», используемый в LabVIEW, основан на архитектуре потоков данных. Последовательность выполнения операторов в таких языках определяется не порядком их следования (как в императивных языках программирования), а наличием данных на входах этих операторов. Операторы, не связанные по данным, выполняются параллельно в произвольном порядке.

Экранная форма интерфейса LabVIEW представлена на рисунке 2.

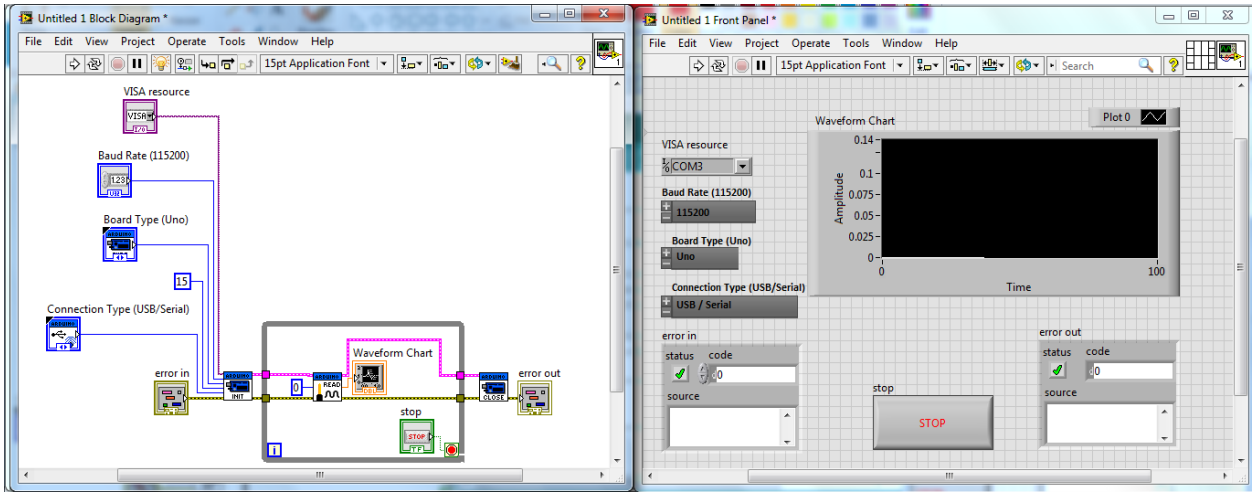


Рисунок 2 – Экранная форма интерфейса LabVIEW

Среда разработки LabVIEW обладает следующими достоинствами:

- разграничение интерфейсов пользователя и разработчика;
- возможность расширения стандартной библиотеки инструментов;
- поддержка платформы Microsoft Windows;
- наличие визуального языка программирования.

Среда разработки LabVIEW имеет следующие недостатки:

- высокая стоимость (базовая версия стоит 400 долларов в год);
- сложности при интерпретации и разработке программ при использовании визуального языка программирования;
- ориентация на разработку систем контрольно-измерительных приборов.

Последней версией среды разработки является LabVIEW 2020.

1.4.3 Программный комплекс JMCAD

JMCAD (JMCADRTS, JMCADRTC) – это программный комплекс для моделирования и симуляции (анализа динамики и проектирования) сложных динамических систем.

Программный комплекс имеет три отдельных независимых блока JMCAD, JMCADRTS, JMCADRTC. JMCAD – основной блок для создания и редактирования моделей. JMCADRTS – блок для запуска модели в режиме работы. JMCADRTC – блок для запуска интерфейса контроля и управления моделью.

Экранная форма интерфейса JMCAD представлена на рисунке 3.

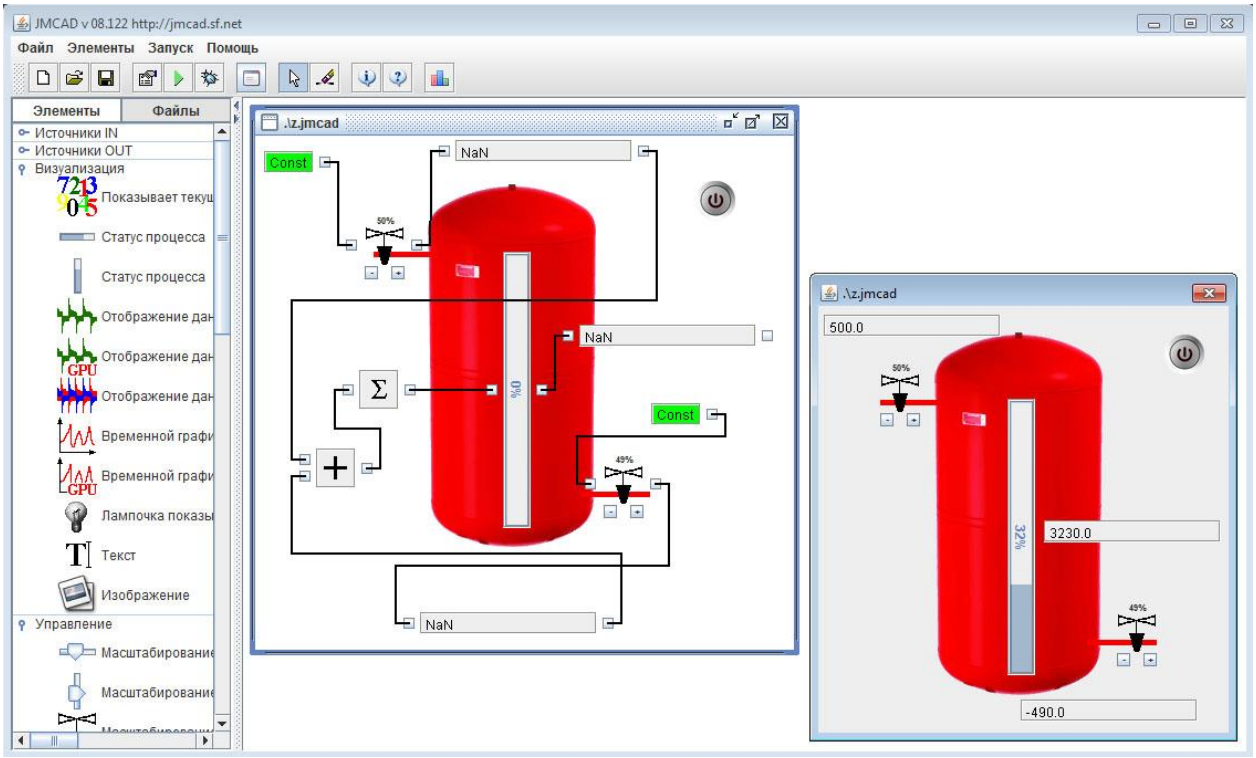


Рисунок 3 – Экранная форма интерфейса JMCAD

Программный комплекс JMCAD может быть рассмотрен как программа, имеющая следующие достоинства:

- бесплатность;
- разграничение интерфейсов пользователя и разработчика;
- возможность расширения стандартной библиотеки элементов за счет написания своих;
- мультиплатформенность;
- встроенный визуальный язык программирования, представленный в виде соответствующих элементов стандартной библиотеки;

• наличие необходимых средств для разработки обучающих программных систем.

К недостаткам программного комплекса JMCAD относятся ограниченные возможности визуального языка программирования.

Последней версией программного комплекса является JMCAD-09.130.

1.5 Выводы к разделу 1

В данном разделе в ходе анализа предметной области были разобраны понятия виртуального лабораторного стенда и САПР для ВЛС, а также рассмотрены особенности виртуальных лабораторных стендов.

В процессе определения задач проекта было решено в качестве подготовительной работы рассмотреть аналоги для определения особенностей разрабатываемого модуля, затем спроектировать модуль проектирования ВЛС и непосредственно реализовать модуль в рамках поставленной цели; оценить экономические затраты, выручку, а также прибыль от реализации модуля проектирования.

Для комплексного рассмотрения аналогов программной реализации была составлена следующая таблица (таблица 1).

Таблица 1 – Сравнение аналогов

Критерий сравнения	Simulink	LabVIEW	JMCAD
1	2	3	4
Стоимость	275 долларов в год; 550 долларов - бессрочно	400 долларов в год	бесплатная
Разграничение интерфейсов пользователя и разработчика	имеется, но требуется знание языка MATLAB	имеется	имеется
Возможность расширения стандартной библиотеки	имеется	имеется	имеется
Целевые платформы	Microsoft Windows, Linux, macOS	Microsoft Windows	мультиплатформенная

Продолжение таблицы 1

1	2	3	4
Использование визуальных языков программирования	имеется, но возможности ограничены	имеется, но большие схемы сложны для восприятия	имеется, но возможности ограничены
Ориентация на создание обучающих программных систем	имеется	в первую очередь ориентирована на разработку контрольно-измерительных систем	имеется

Изучение аналогов показало, что требуется реализовать модуль, который поддерживал бы изначальное разделение модели на внутреннее логическое представление работы модели и внешнее представление, видимое пользователю при интерпретации или выполнения исполняемого файла модели; имел визуальный язык программирования, удобный в использовании, который бы позволял организовывать не только работу на уровне математических моделей, но и был способен в целом организовывать работу модели как самостоятельной программы, способной к взаимодействию с пользователем; имел более низкую цену по сравнению с аналогами.

2 Проектирование модуля проектирования виртуальных лабораторных стендов

После принятия решения о необходимости разработки модуля следует произвести проектирование структуры программы. Помимо разработки функциональных моделей и объектно-ориентированных моделей на языке UML, определения структуры данных для сохранения и загрузки модели, в настоящей работе имеет большое значение формирование концепции модели и описание визуального языка программирования, с помощью которого будет осуществляться процесс разработки модели в данной САПР.

2.1 Определение концепции модели

Чтобы перейти к проектированию модуля, необходимо задаться вопросом о том, что же представляет собой модель в рамках данного модуля. Конечно, ранее уже давалось замечание о том, что понятия «модель» и «виртуальный лабораторный стенд» являются синонимами, а определение последнего было дано в анализе предметной области. Однако погружение в архитектуру модуля проектирования виртуальных лабораторных стендов требует более детального понимания модели.

Модель – это совокупность связанных тем или иным способом объектов модели, имеющих список привязанных переменных объекта, ориентированных на взаимодействие с объектами модели и пользователем через задаваемое с помощью визуального языка программирование поведение, предназначенная для моделирования процесса, явления или объекта конкретной предметной области.

Именно на создание модели с последующим ее сохранением в виде файла специального формата ориентирован весь процесс проектирования в данном модуле. Дальнейшие действия по работе с моделью предполагают либо продолжение редактирования содержимого модели с целью перевода ее в состояние, при котором пользователь модуля сочтет уровень соответствия модели своим ожиданиям допустимым, либо передача файла модели другим модулям для интерпретации или компиляции.

Следует отметить, что процесс выполнения модели имеет в своей основе понятие цикла работы модели.

Перед рассмотрением цикла работы модели важно еще раз обратиться к определению модели и разобрать на более детальном уровне ключевые компоненты модели: «объект модели», «переменная объекта», «поведение объекта».

2.1.1 Объект модели

Объект модели (объект) – это семантически и программно обособленная единица модели, несущая в себе часть информации в форме переменных объекта и часть логики работы модели в форме поведения объекта.

Объект модели содержит также служебную информацию: уникальный идентификатор объекта, изображение объекта, название объекта, список ассоциаций объекта, контейнер логики, список связанных объектов, базовый тип объекта, флаги активности, редактирования переменных и поведения объекта, заголовок объекта. В настоящий момент не нужно вдаваться в смысл каждого из элементов служебной информации объекта, необходимо лишь иметь ввиду, что такие понятия в рамках данной САПР существуют и будут раскрываться по мере необходимости.

Каждый объект модели состоит из нескольких частей, состав которых – разный, в зависимости от типа объекта. Можно выделить следующий общий вид объекта модели по составу частей (рисунок 4).

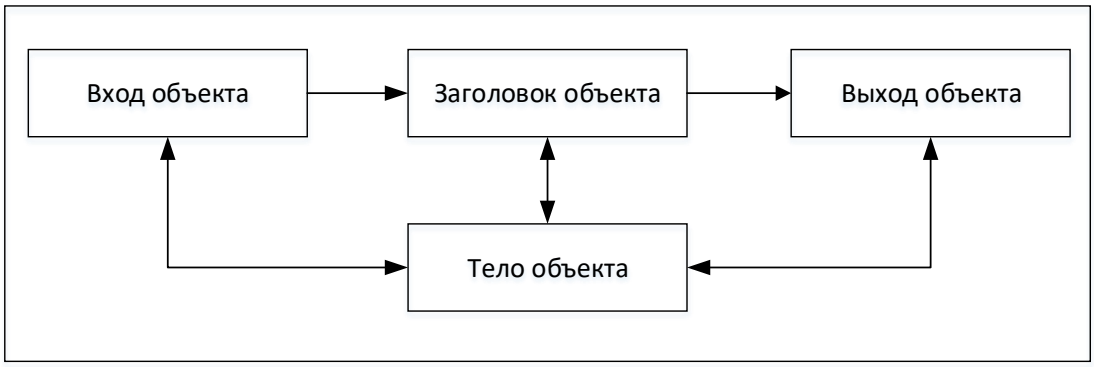


Рисунок 4 – Общий вид объекта по составу частей

Следует сразу отметить, что на рисунке изображены только части, которые имеют наибольшее значение для рассмотрения в рамках данного пункта.

Как видно из рисунка 4, объект в своем общем виде состоит из четырех частей, которые связаны линиями передачи данных. Вход объекта содержит входные переменные, и на него поступает информация от других объектов. Заголовок объекта содержит переменные общего назначения для хранения промежуточных значений, полученных в результате обработки данных в объекте, часть логики в форме программ, связанных с событиями: запуска модели и обновления состояния модели. Выход объекта содержит выходные переменные, и с него информация поступает на входы других

объектов. Тело объекта содержит список публичных функций, доступных для исполнения самим объектом и связанными с ними объектами.

Связь между объектами подразумевает, что объекты могут передавать данные между своими входами и выходами, а также вызывать публичные функции друг друга, и в графической форме представлена в виде линии. Также связь в рамках данной САПР называется ассоциацией.

Всего существует четыре типа объектов, которые называются базовыми типами объектов: объект общего назначения, объект ввода, объект ввода, объект вывода, объект данных.

Первый тип объекта соответствует по составу рисунку 4. Он непосредственно не взаимодействует с пользователем, а, как и объект данных, взаимодействует только с другими объектами.

Объекты ввода и вывода предназначены не только для взаимодействия с другими объектами, но и получения данных от обучающегося и передачи ему результатов обработки за счет того, что помимо контейнера логики имеют и контейнер интерфейса.

Контейнеры представляют собой графическое представление объекта на поле проектирования, и с помощью их пользователь модуля проектирования может взаимодействовать с объектами. Всего существует два типа контейнеров: контейнер логики и контейнер интерфейса. Контейнер логики является основным контейнером объекта модели, с помощью него пользователь модуля изменяет список переменных объекта и задает поведение объекту. Контейнер интерфейса предназначен для отображения объекта в виде некоторого элемента управления, называемого объектом интерфейса, расположенного на представлении интерфейса стенда, с которым уже может взаимодействовать обучающийся.

Объект интерфейса может содержать программы, обрабатывающие события, специфичные для данного объекта (например, событие «Клик» для объекта интерфейса «Кнопка»).

Состав частей объектов ввода и вывода представлен на рисунках 5 и 6 соответственно.

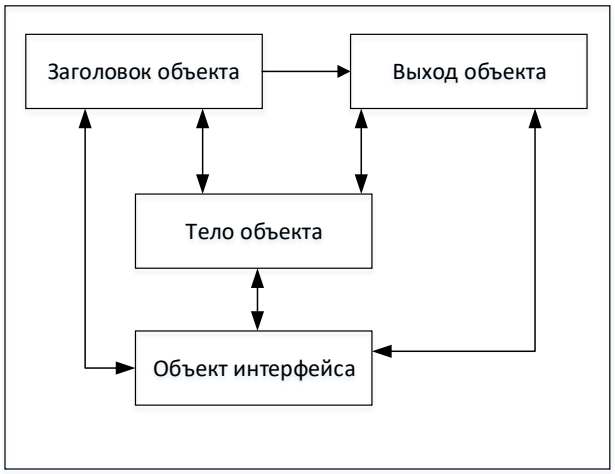


Рисунок 5 – Состав частей объекта ввода

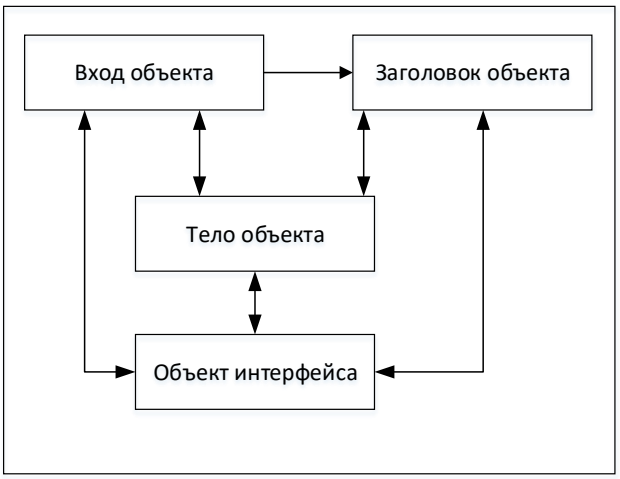


Рисунок 6 – Состав частей объекта вывода

Объект данных предназначен исключительно для хранения данных, поэтому имеет только заголовок объекта, но при этом он автоматически с момента создания в неявном виде связан со всеми объектами модели.

Базовые типы объектов не могут использоваться сами по себе, а на их основе должны быть уже созданы объекты, входящие в состав определенных расширений, – объекты расширений (рисунок 7).

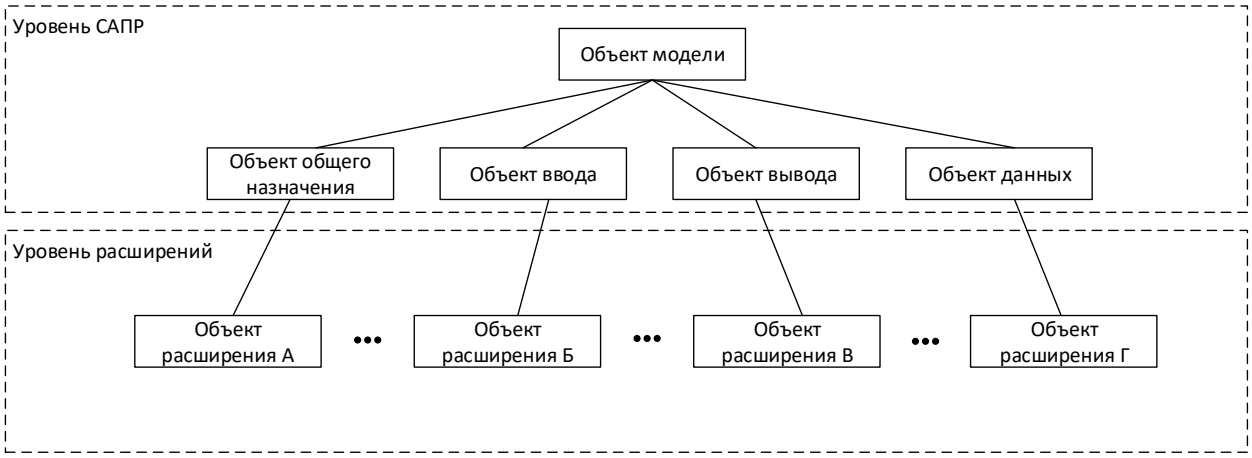


Рисунок 7 – Два уровня реализации объектов в данной САПР

Именно за счет расширений, которые представляют собой совокупность объектов модели конкретной предметной области, возможно создание различных по своей направленности моделей.

Однако сразу следует обратить внимание на то, что объекты расширений можно называть объектами модели и одним из базовых типов объектов, соответствующих данному объекту расширений, с тем замечанием, что они имеют некоторые дополнительный функционал, позволяющий отнести объект модели к конкретной предметной области.

2.1.2 Переменная объекта

Вторым базовым понятием, после «объект модели», является такой элемент данного САПР, как «переменная объекта».

Переменная объекта (переменная) – это программный контейнер, инкапсулирующий в себе значение данной переменной с дополнительной служебной информацией для корректной обработки переменной в контексте модели и для обеспечения взаимодействия с ней пользователя, и содержащий в себе совокупность методов по выполнению служебных операций с переменной.

В зависимости от типа переменной совокупность служебной информации и методов может быть различной, но для всех типов переменных характерен следующий список метаинформации о переменной: название переменной, описание переменной, тип данных переменной.

Классификация переменных представлена на рисунке 8.

Рисунок 8 – Классификация переменных объекта

По типу создания переменные делятся на программные и пользовательские. Программные переменные создаются разработчиком расширений и изначально внедрены в программный код объектов расширений, поэтому не могут быть удалены, и изменять можно лишь само значение переменной объекта, а в некоторых случаях – и оно доступно только для чтения. Пользовательские переменные создаются пользователем во время разработки модели и могут быть редактироваться и удаляться в процессе работы над моделью. Причем пользователь может изменять часть служебной информации о пользовательской переменной: название переменной и ее описание.

По типу данных переменные делятся на логические (принимаются значения True или False), целые, дробные, текстовые, содержащие текстовый файл и содержащие файл изображения. Последние два типа хранят путь к файлу формата «.txt» и форматов «.jpg», «.bmp», «.png» соответственно относительно расположения файла модели.

По назначению переменные делятся на переменные интерфейса и переменные логики. Первые всегда относятся к программным переменным и расположены в заголовке объекта. Они необходимы для задания свойств объектам интерфейса. Вторые переменные расположены в одной из трех частей объекта модели: вводе, выводе или заголовке объекта. Они используются для формирования логики работы модели.

Переменные логики также делятся по направлению передачи данных на переменные общего назначения, переменные ввода и переменные вывода. Отдельного внимания здесь заслуживают переменные ввода и вывода, так как их механизм работы отличается от механизма работы переменной общего

назначения. Если переменная общего назначения предназначена для хранения промежуточных данных объекта, то другие два типа переменных предназначены для реализации обмена данных между объектами и представляют собой ссылочные типы данных. Ссылочный характер переменной вывода проявляется в том, что при задании ей значения, это значение будет задано для всех переменных ввода, связанных с ней. В свою очередь, ссылочный характер переменных ввода проявляется в том, что они связаны с переменной вывода, и их значение является копией значения соответствующей переменной вывода, за исключением случаев явного вмешательства в работу механизма связей с помощью поведения объекта. Переменным ввода нельзя явно задать значение, они доступны в этом контексте только для чтения. Одна переменная ввода может ссылаться только на одну переменную вывода, с другой стороны, одна переменная вывода может иметь несколько связанных с ней переменных ввода. Причем обе эти переменные должны относиться в случае связи к разным объектам модели. Также важно заметить, что связать можно лишь переменные, типы данных которых совпадают.

2.1.3 Поведение объекта

После введения переменной объекта необходимо каким-то образом предоставить пользователю возможность формировать логику работы модели на уровне отдельных объектов. Здесь добавляется понятие «поведение объекта».

Поведение объекта – совокупность программ объекта модели, которые реализуют определенные алгоритмы обработки полученных объектом данных, сохранения результатов обработки и (или) передачи их другим объектам и пользователю.

Программа – это последовательность конструкций нижнего уровня визуального языка программирования LabScript, LabScript for Behaviour. Подробнее о языке будет рассмотрено в пункте «Визуальный язык программирования LabScript».

Каждая программа привязана к определенному событию, обработкой которого она занимается. Всего существует два типа событий: системные и события интерфейса. Системные события представлены всего двумя событиями: «Старт», которое генерируется для каждого объекта модели при ее запуске на исходном большом цикле работы модели, «Обновление», которое генерируется для каждого объекта в ходе промежуточного большого цикла работы модели. Программы, обрабатывающие данные события, расположены в заголовке объекта модели. События интерфейса являются специфичными для каждого объекта модели и реализованы в объекте интерфейса, связанного с данным объектом. Соответственно, там же и

располагаются программы для обработки этих событий. Подробнее о цикле работы модели и его видах рассказано в следующем пункте.

2.1.4 Цикл работы модели

Выполнение модели модулем интерпретации или как отдельного исполняемого файла связано с циклом работы модели.

Цикл работы модели – строго заданная последовательность выполнения программ объектов модели.

В связи с тем, что были выделены два типа событий, можно выделить и два типа циклов работы модели: большой цикл работы модели, связанный с обработкой системных событий, малый цикл работы модели, связанный с обработкой событий интерфейса. Если больших циклов работы модели может быть только два, то малых циклов может быть столько же, сколько и объектов интерфейса имеется у модели.

Схематично оба эти типа циклов можно изобразить следующим образом (рисунок 9).

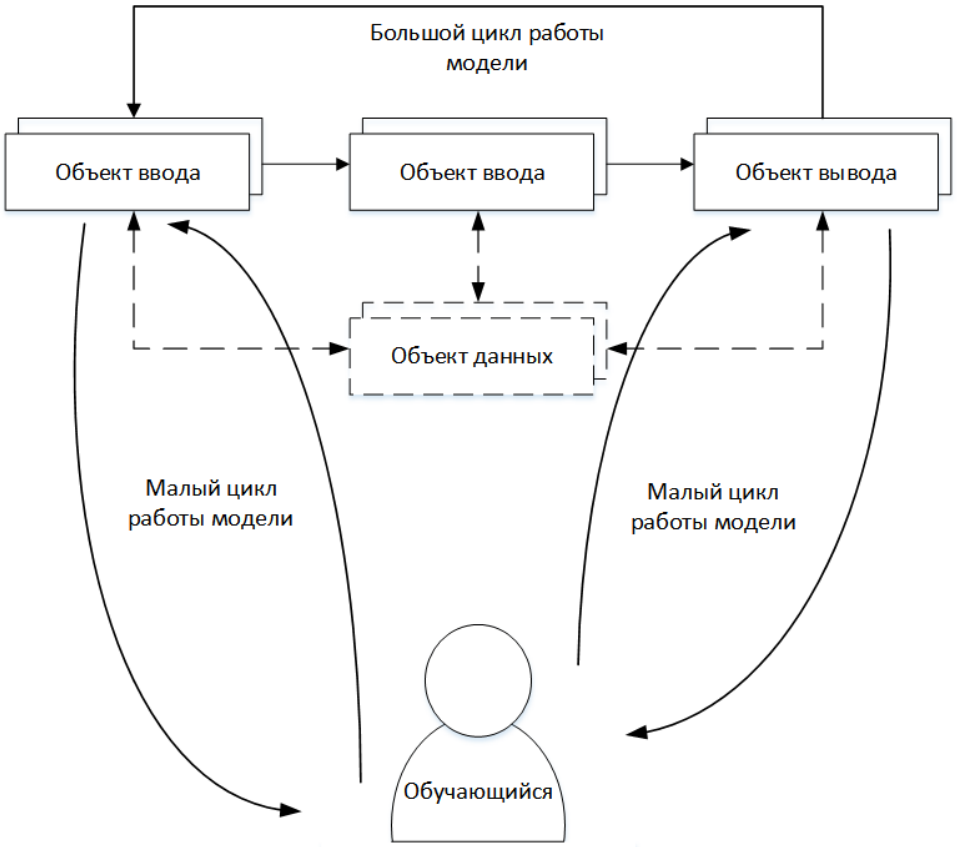


Рисунок 9 – Циклы работы модели

На рисунке 9 сплошными линиями показаны передача управления, а пунктирными – обмен данными. Под передачей управления от объекта к объекту подразумевается обращение к объекту для выполнения его

программ, соответствующих типу большого цикла работы модели. Передача управления от пользователя к объекту заключается во взаимодействии пользователя с объектом интерфейса, что приводит к генерации соответствующего события и вызова программы обработки данного события. Под передачей управления от объекта к пользователю подразумевается завершение обработки события, связанного с объектом интерфейса, и предоставлению пользователю возможности следующего взаимодействия с объектами интерфейса. Следует рассмотреть подробнее каждый из циклов работы модели.

Начать рассмотрение можно с большого цикла работы модели, как основополагающего. Существует строгая последовательность передачи управления: первым делом происходит выполнение программ, содержащихся в объектах ввода, далее – в объектах общего значения, и, в конечном счете, – в объектах вывода. Это связано логически с тем, что данные на начальном этапе проходят через процесс получения, затем – через процесс обработки, результаты которого передаются процессу отправки результатов обработки данных. Именно такой последовательности подчиняется большой цикл работы модели. Исключением из данного правила являются объекты данных, которые непосредственно не участвуют в циклах работы модели, однако к ним могут обращаться другие объекты в ходе получения управления для сохранения или получения данных. Выделяются два типа больших циклов работы модели: исходный большой цикл работы модели и промежуточный большой цикл выполнения модели. Первый предназначен для выполнения программ, связанных с системным событием «Старт» при запуске модели на исполнение. Промежуточный большой цикл работы модели предназначен для выполнения программ, связанных с системным событием «Обновление» при итерационном характере работы модели после выполнения исходного большого цикла до окончания процесса выполнения модели.

Отдельным видом циклов работы модели выступает малый цикл работы модели. Его особенность в том, что он выполняется независимо от большого цикла работы модели и для того объекта модели, у которого было сгенерировано событие объекта интерфейса. То есть цикл начинается с того, что пользователь передает управление объекту интерфейса, далее после выполнения соответствующей программы управление передается пользователю, и цикл завершается.

2.2 Разработка функциональных моделей

В качестве следующего этапа проектирования модуля выступает разработка функциональных моделей с использованием методологий IDEF0, IDEF3, DFD.

Контекстная диаграмма IDEF0 представлена в приложении А, лист 82.

На контекстной диаграмме в качестве входов в блок модуля выступают файл модели при его наличии, описание предметной области ВЛС, файлы расширений, необходимые для реализации ВЛС в рамках указанной предметной области.

В качестве управления выступают инструкции по эксплуатации модуля и требования, которыми должна соответствовать ВЛС, чтобы поддерживалась целостность и непротиворечивость объектов ВЛС.

В качестве механизмов выступают непосредственно пользователь модуля и программная реализация модуля.

На выходе в процессе проектирования ВЛС можно получить файл модели.

Декомпозиция контекстной диаграммы IDEF0 представлена в приложении А, лист 83.

На диаграмме декомпозиции можно видеть, что процесс проектирования состоит из трех подпроцессов: подготовки рабочего пространства, непосредственно процедуры проектирования ВЛС и сохранения результатов.

Подготовка рабочего пространства (приложение А, лист 84) включает в себя подключение необходимых для процесса проектирования расширений и подготовку модели, то есть загрузку модели, либо создание новой при отсутствии файла модели.

Процедура проектирования ВЛС (приложение А, лист 85) состоит из трех подпроцессов, необходимых для разработки ВЛС: проектирования содержимого ВЛС, изменение списка расширений и изменения информации о ВЛС – и подпроцесса оценки качества созданной модели с точки зрения пользователя, при котором возможно возвращение на предыдущие подпроцессы для исправления найденных замечаний.

Отдельно следует сказать о подпроцессе проектирования содержимого ВЛС (приложение А, лист 86). Данный подпроцесс декомпозируется на четыре стадии: задания состава объектов ВЛС путем удаления и добавления новых объектов расширений на поле проектирования логики, изменения интерфейсной части ВЛС, которую будет видеть обучающийся в процессе выполнения модели, установления связей между объектами, задания (изменения) переменных объектов и изменения поведения объектов.

Подпроцесс сохранения результатов (приложение А, лист 87) необходим для формирования модели из совокупности объектов модели и информации о ВЛС, которая потом сохраняется в виде файла.

Контекстная диаграмма IDEF3 представлена в приложении Б, лист 89.

На диаграмме декомпозиции (приложение Б, лист 90) представлено три идущих последовательно элемента поведения: подготовка рабочего пространства, проектирование ВЛС, сохранение результатов.

Подготовка рабочего пространства (приложение Б, лист 91) включает последовательное выполнение двух элементов поведения: подключения расширений и подготовки модели.

Проектирование ВЛС (приложение Б, лист 92) происходит следующим образом. Начинается процесс проектирования содержимого модели, после которого осуществляется оценка качества проектирования: в случае наличия недочетов происходит возвращение к началу проектирования с изменением списка подключенных расширений и информации о ВЛС при необходимости.

Проектирование содержимого ВЛС (приложение Б, лист 93) представляет собой последовательность элементов поведения: определение списка объектов, изменение интерфейса ВЛС, установление связей между объектами, изменение переменных объектов и изменение поведения объектов.

При сохранении результатов (приложение Б, лист 94) сначала формируется модель, далее происходит сохранение модели в файл.

Контекстная диаграмма DFD представлена в приложении В, лист 96. Модель модуля DFD представляет собой обобщенное отображение архитектуры программной реализации модуля.

На контекстной диаграмме DFD отображено взаимодействие модуля проектирования ВЛС с пользователем и с внешним хранилищем, под которым подразумевается в общем случае энергонезависимый носитель информации.

Модуль проектирования ВЛС разбит на несколько менеджеров (приложение В, лист 97): менеджер модели, менеджеров проектирования, менеджер расширений, менеджер строки состояния, менеджер консоли. Менеджер представляет собой бизнес-логику, отделенную от интерфейса. Особое внимание заслуживает блок «Элементы управления», который объединяет в себе всю совокупность элементов пользовательского интерфейса со специфичной для них логикой обработки действий пользователя. В элементы управления входит и механизм проектирования поведения объекта. Также в ходе проектирования используются следующие хранилища: хранилище с информацией о ВЛС, хранилище со списком объектов и хранилище с объектами ввода и вывода.

Менеджер модели предназначен для осуществления процесса загрузки, сохранения файла модели, создания новой модели путем указания информации о ВЛС, изменения информации о ВЛС.

Менеджер расширений предназначен для подключения и отключения расширений, а также для передачи описаний типов объектов элементам интерфейса для их создания. Кроме того, он позволяет посмотреть информацию о расширении. Имеет хранилище подключенных расширений.

Менеджеры проектирования (приложение В, лист 98) предназначены для осуществления непосредственно процесса проектирования. Они состоят из следующих менеджеров: менеджер поля проектирования логики, менеджер поля проектирования интерфейса, менеджер переменных логики.

Менеджер поля проектирования логики предназначен для организации логики работы модели через организацию взаимосвязей между объектами и определение состава данных объектов.

Менеджер поля проектирования интерфейса предназначен для расположения объектов интерфейса для формирования дизайна интерфейса ВЛС.

Менеджер переменных логики предназначен для редактирования переменных логики объектов.

Менеджер консоли (приложение В, лист 97) предназначен для формирования обратной связи с пользователем через консоль.

Менеджер строки состояния (приложение В, лист 97) предназначен для формирования обратной связи с пользователем через строку состояния.

2.3 Разработка моделей данных

Модели данных являются неотъемлемой частью любой информационной системы. В рамках данной работы все модели данных можно разделить на две большие группы: модели данных, используемые при работе модуля, и модели данных, используемые для сохранения и загрузки модели.

Первая группа моделей данных непосредственно связана с классами, поэтому будет отображена на диаграмме классов.

Вторая группа моделей данных является производной от первой группы моделей данных с тем отличием, что эти данные пригодны для сохранения и загрузки на энергонезависимый носитель информации.

В ходе выбора способа хранения модели было принято решение разработать собственный формат хранения данных в связи с тем, чтобы

повысить скорость процесса сохранения и загрузки данных и управлять данными процессами на более низком уровне.

Основой для реализации вышеупомянутого стало использование механизма бинарной сериализации и десериализации, предоставляемого платформой .NET Framework, с дополнением его разработанным в рамках выпускной квалификационной работы механизма восстановления ссылочной целостности данных, о котором будет сказано при проектировании функции «Загрузить модель» в пункте 2.5.

Диаграмма сериализации моделей данных представлена в приложении Г.

Всю структуру сериализуемых данных можно описать с точки зрения иерархии.

Коренным объектом иерархии сериализации является модель, которая включает в себя метаданные о модели и совокупность объектов расширений.

Объекты расширений имеют некоторый базовый набор данных, содержащихся в каждом объекте модели. К ним относятся идентификационный номер объекта, имя объекта, базовый тип объекта, идентификационный номер расширения, с которым связан данный объект, флаги редактирования алгоритма поведения и переменных объекта, состояние активности, список ассоциаций, контейнер логики объекта и заголовок объекта.

Ассоциации хранят в себе идентификационные номера связанных объектов, логическую переменную, указывающую на наличие линии, и линию, при ее наличии. Линия хранится только у одной ассоциации из пары для избегания дублирования.

Линия хранит в себе: масштаб линии, начальную и конечную точки линии в оригинальном масштабе (масштабе по умолчанию), список точек для создания пути между объектами.

Контейнер логики объекта является производным от контейнера. Контейнер в общем случае хранит следующие данные: идентификационные номер родительского объекта, к которому он прикреплен, объект интерфейса, который оборачивается данным контейнером, масштаб, размер и положение при оригинальном масштабе.

Объект интерфейса должен быть пригодным для сохранения и загрузки. Разработчик может реализовать свой алгоритм сохранения и загрузки объекта интерфейса или воспользоваться готовым классом, позволяющим сохранить размеры и положение объекта интерфейса.

Заголовок объекта хранит информацию об уникальном идентификаторе объекта модели, к которому он привязан, программы обработки событий «Старт» и «Обновление», список переменных общего назначения.

Помимо заголовка, в зависимости от базового типа объекта, объект модели может иметь и другие части: вход объекта и выход объекта, для которых сохраняются списки выходных и выходных переменных соответственно.

Переменная объекта в общем случае хранит следующие данные: уникальный идентификатор переменной, имя переменной, краткое описание переменной, уникальный идентификатор объекта модели, с которой она связана, направление передачи данных, тип данных переменной, данные переменной. Для входной переменной также хранится информация о наличии выходной переменной, с которой она связана, в виде уникального идентификатора, при наличии связанной выходной переменной, и логического значения, указывающего на наличие связи. Для выходной переменной – логическое значение, отображающее непустое состояние списка уникальных идентификаторов входных переменных, связанных с ней, и сам список, при его наличии.

Программа хранит в себе последовательность конструкций. В зависимости от типа конструкций они хранят различный набор данных: конструкция присваивания хранит выражение операнда переменной, которой присваивается новое значение, и выражение, значение которого присваивается; конструкция вызова хранит в себе выражение операнда вызова; управляющая конструкция содержит тип управляющей конструкции (условие «ЕСЛИ», подусловие «ИЛИ_ЕСЛИ», подусловие «ИНАЧЕ» или цикл «ПОКА»), условие выполнения конструкции в виде логического выражения и список конструкций.

Выражение в общем случае хранит информацию о типе дополнительной функции и типе данных выражения. Арифметическое выражение хранит информацию о типах арифметических операций и список самих операндов. Логическое выражение хранит тип логической операции и список логических операндов.

Выражения операнда делятся на три типа, в зависимости от которого формируется разный набор сохраняемых данных: выражение операнда константы, выражение операнда вызова и выражение операнда переменной. Выражение операнда константы хранит информацию о типе данных константы и значение самой константы. Выражение операнда вызова хранит информацию о вызываемой функции. Функция обладает следующим набором данных для сохранения: название функции, краткое описание

функции, возможность возвращения значения, указывающая на наличие или отсутствие возвращаемого типа данных, возвращаемый тип данных, списки имен и типов данных аргументов.

Выражение операнда переменной в общем случае хранит информацию о том, доступно ли значение переменной только для чтения. Выделяются два типа выражений операнда переменной: выражение операнда логической переменной и выражение операнда переменной интерфейса. Выражение операнда логической переменной хранит информацию о переменной и объекте в виде уникальных идентификаторов, направлении передачи данных переменной. Выражение операнда переменной интерфейса содержит уникальный идентификатор объекта и название свойства, с которым связан операнд.

2.4 Визуальный язык программирования LabScript

Ранее упоминалось, что одним из ключевых компонентов модели является поведение объекта.

Реализация его возможна за счет нижнего уровня разработанного визуального языка программирования под названием LabScript. Всего у данного языка имеется два уровня реализации: верхний и нижний.

Верхний уровень называется LabScript for Objects и представляет собой подмножество языка программирования для манипуляции объектами модели: созданием и удалением их, организацией системы связей. В связи с тем, что модуль может иметь в своем составе различный список подключенных расширений, то и состав графических объектов верхнего уровня языка может существенно различаться. Именно под этим верхним уровнем подразумевается тот механизм работы с объектами, о котором в том или ином виде было рассказано в других разделах выпускной квалификационной работы. Больший интерес с учетом вышесказанного в рамках данного пункта имеет нижний уровень визуального языка программирования LabScript.

Нижний уровень называется LabScript for Behaviour и предназначен для формирования программ для обработки событий в объекте модели. Если верхний уровень языка представляет собой совокупность графических элементов, то нижний уровень визуального языка программирования имеет в своем составе уже текстографические элементы. Это означает, что при проектировании программ на LabScript for Behaviour возрастает роль текстовых обозначений, но при этом ими можно манипулировать, как графическими элементами.

2.4.1 Синтаксис LabScript for Behaviour

Имеет смысл записать упрощенный синтаксис подмножества языка в расширенной форме Бэкуса-Наура:

Программа = {Конструкция}.

Конструкция = Управляющая конструкция | Конструкция вызова | Конструкция присваивания.

Управляющая конструкция = Цикл | Условие.

Условие = Условие «ЕСЛИ», {Условие «ИЛИ_ЕСЛИ»}, [Условие «ИНАЧЕ»].

Условие «ЕСЛИ» = «ЕСЛИ », Логическое выражение, « , ТО:», {Конструкция}.

Условие «ИЛИ_ЕСЛИ» = «ИЛИ_ЕСЛИ », Логическое выражение, « , ТО:», {Конструкция}.

Условие «ИНАЧЕ» = «ИНАЧЕ:», {Конструкция}.

Цикл = «ПОКА », Логическое выражение, « , ДЕЛАТЬ:», {Конструкция}.

Конструкция вызова = (Дополнительная функция, «(», Выражение операнда вызова, «)») | Выражение операнда вызова.

Конструкция присваивания = Выражение операнда переменной, « = », (Выражение операнда | Арифметическое выражение).

Арифметическое выражение = (Дополнительная функция, «(», Исходное арифметическое выражение, «)») | Исходное арифметическое выражение.

Исходное арифметическое выражение = «(», (Арифметический операнд, Арифметическая операция){(Арифметический операнд, Арифметическая операция)}, Арифметический операнд, «)».

Дополнительная функция = «ЭКСПОНЕНТА» | «КВАДРАТ» | «КОРЕНЬ» | «ЛОГ-10» | «ЛОГ-N» | «СИНУС» | «КОСИНУС» | «ТАНГЕНС» | «КОТАНГЕНС» | «ЦЕЛОЕ» | «ДРОБНОЕ» | «ТЕКСТ».

Арифметический операнд = Арифметическое выражение | Выражение операнда.

Арифметическая операция = « + », « - », « / », « * ».

Логическое выражение = (Дополнительная функция, «(», Исходное логическое выражение, «)») | Исходное логическое выражение.

Исходное логическое выражение = «(», (Выражение, Упрощенная логическая операция, Выражение) | (Группа операндов «И» | Группа

операндов «ИЛИ» | Группа операндов «>» | Группа операндов «<» | Группа операндов «>=» | Группа операндов «<=»), «)».

Логический операнд = Логическое выражение | Выражение операнда.

Группа операндов «И» = (Логический операнд, « И »){ (Логический операнд, « И »)}, Логический операнд.

Группа операндов «ИЛИ» = (Логический операнд, « ИЛИ »){ (Логический операнд, « ИЛИ »)}, Логический операнд.

Группа операндов «>» = (Логический операнд, « > »){ (Логический операнд, « > »)}, Логический операнд.

Группа операндов «<» = (Логический операнд, « < »){ (Логический операнд, « < »)}, Логический операнд.

Группа операндов «>=» = (Логический операнд, « >= »){ (Логический операнд, « >= »)}, Логический операнд.

Группа операндов «<=» = (Логический операнд, « <= »){ (Логический операнд, « <= »)}, Логический операнд.

Упрощенная логическая операция = « == », « != ».

Выражение операнда = (Дополнительная функция, «(», Исходное выражение операнда, «)») | Исходное выражение операнда.

Исходное выражение операнда = Выражение операнда константы | Выражение операнда вызова | Выражение операнда переменной.

Выражение операнда константы = Целое число | Дробное число | Логическое значение | Текст | Файл | Изображение.

Логическое значение = «True» | «False».

Текст = «”», Строка, «”».

Файл = «[txt]”», Путь к текстовому файлу, «”».

Изображение = «[img]”», Путь к изображению, «”».

Выражение операнда вызова = Название объекта, «.», Название функции, «(», Список аргументов, «)».

Список аргументов = [{Выражение операнда, «, »}, Выражение операнда].

Выражение операнда переменной = Название объекта, «.», Название переменной.

Назначение выше представленного синтаксиса нижнего уровня языка заключается в том, что он используется для формирования правил построения элементов подмножества языка и для представления программ,

хранящихся в оперативной памяти, в текстовом виде, пригодном для восприятия их пользователем.

Упрощенным данный синтаксис можно назвать потому, что некоторые нетерминалы, указанные в РБНФ, не раскрываются явным образом. Причина этого в том, что данные нетерминалы реализуются на более низком уровне, чем логика работы визуального языка программирования, за счет средств языка программирования C#. Для каждого нераскрытого нетерминала будет дано пояснение в следующем абзаце.

Всего существует восемь нетерминалов, для которых нет описания в вышеуказанном синтаксисе: «Целое число», «Дробное число», «Текст», «Путь к текстовому файлу», «Путь к изображению», «Название объекта», «Название переменной», «Название функции». Нетерминал «Целое число» соответствует объекту типа Int32 языка программирования C#; «Дробное число» – объекту типа Double языка программирования C#; «Текст» – объекту типа String языка программирования C#; «Путь к текстовому файлу» – объекту типа String языка программирования C#, при этом строка является валидной с точки зрения относительного пути к файлу формата «.txt»; «Путь изображению» – объекту типа String языка программирования C#, при этом строка является валидной с точки зрения относительного пути к файлам форматов «.bmp», «.png», «.jpg»; «Название объекта», «Название переменной» и «Название функции» – объектам типа String языка программирования C# и обозначают соответствующие названия объектов модели, переменных объектов, функций объекта. Под функциями объекта подразумеваются публичные функции объекта, доступные для использования в поведении объекта.

Следует заметить, что в синтаксисе языка отсутствуют элементы отделения конструкций друг от друга, так как данное подмножество фигурирует как подмножество визуального языка программирования, поэтому конструкции изначально отделены друг от друга на программном уровне – хранятся в различных объектах структур данных. Представление программ и их составных частей на нижнем уровне языка в виде программных классов можно увидеть на диаграмме классов, о которой сказано в пункте 2.5.

Еще важным моментом является то, что любая структура подмножества LabScript for Behaviour является наследником класса исполняемого элемента, то есть имеет в своей функциональности метод для запуска механизма исполнения. Данный метод в текущей версии имеет пустое тело и будет реализован в ходе разработки модуля интерпретации.

2.4.2 Обработка ошибок

Все ошибки, которые связаны с программами, созданными на LabScript for Behaviour, можно разделить на ошибки процесса проектирования и ошибки процесса исполнения. Последние могут возникнуть только на этапе исполнения модели, поэтому будут выявляться модулем интерпретации либо в ходе исполнения файла модели в формате «.exe». В настоящей работе будут рассмотрены лишь ошибки процесса проектирования.

Обработка ошибок процесса проектирования, как синтаксических, так и семантических, нижнего уровня языка LabScript осуществляется за счет механизмов, реализованных как в элементах управления, представляющих собой графическое отображение операндов, так и в Мастере действий, связанным непосредственно с проектированием целых конструкций и выражений. Каждый элемент управления операндом генерирует пустое возвращаемое значение операнда в случае найденных ошибок на уровне операнда, в результате Мастер действий имеет информацию об операндах, сгенерировавших ошибку, и об ошибках на уровне конструкции или выражения и принимает конечное решение о списке найденных ошибок исходя из конкретной ситуации.

Для удобства пользователей было принято решение выводить все найденные ошибки в виде отдельного окна в форме списка. Каждая ошибка характеризуется объектом возникновения ошибки, кодом ошибки и текстом ошибки.

Все ошибки, которые могут быть найдены при существующем механизме поиска ошибок, представлены в таблице 2.

Таблица 2 – Классификация ошибок

Объект возникновения ошибки	Код ошибки	Текст ошибки
1	2	3
Конструкция или выражение	1.1	Отсутствуют операнды
	1.2	Конструкция или выражение имеет неправильный формат
	1.3	Должно быть, как минимум, два операнда
	1.4	Типы данных операндов не совпадают
	1.5	Для логических данных доступны только операции 'И' и 'ИЛИ'

Продолжение таблицы 2

1	2	3
Операнд	2.1	Операнд не имеет значения или неправильно задан
	2.2	Тип операнда не соответствует типу конструкции или выражения
	2.3	Функция должна возвращать непустое значение в выражении
	2.4	Первый операнд не может иметь функцию
	2.5	В арифметическом выражении можно использовать только целые и дробные числа
	2.6	В логическом выражении с число операндов, больше 2, можно использовать только логический тип данных
	2.7	Функция должна возвращать непустое значение в качестве аргумента указанной функции
	2.8	Тип данных аргумента не совпадает с указанным
Операция	3.1	Операция имеет неправильный формат
	3.2	Операция не выбрана
	3.3	Для строк доступна только операция сложения
	3.4	Операции не совпадают

2.5 Моделирование с использованием языка UML

Для моделирования модуля проектирования ВЛС были использованы диаграммы вариантов использования, диаграмм активности, последовательности, состояний, классов и компонентов.

Рассмотрение результатов моделирования следует начать с диаграмм вариантов использования, как диаграмм, задающих основные функции, доступные пользователям разрабатываемого модуля.

Контекстная диаграмма модели вариантов использования представлена на рисунке 10.

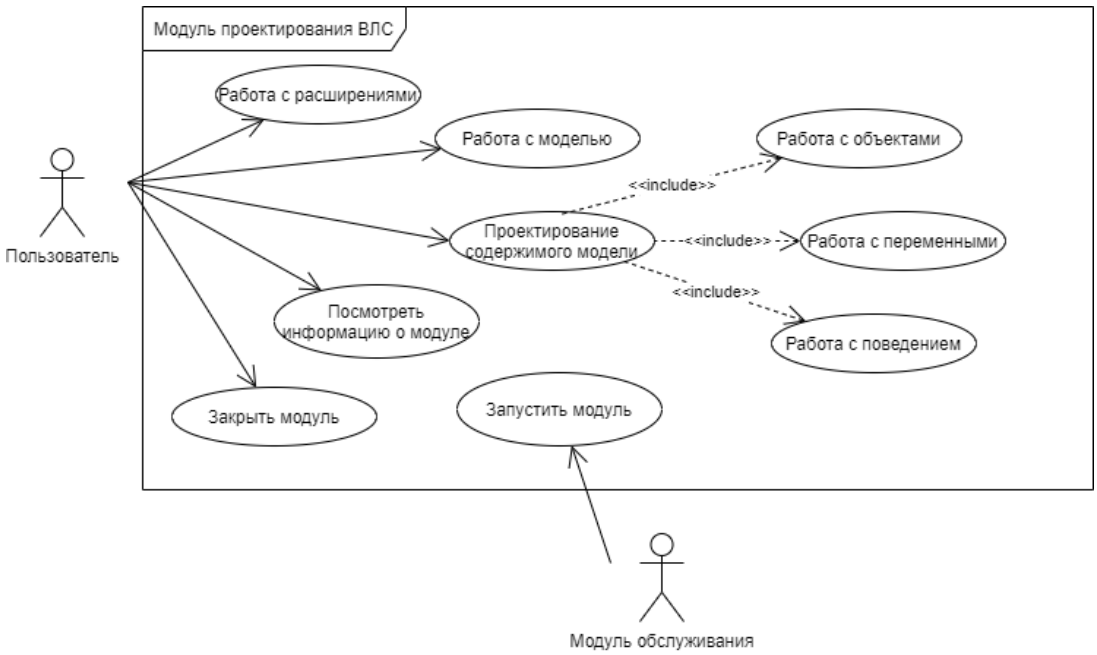


Рисунок 10 – Контекстная диаграмма модели вариантов использования

Актерами модуля проектирования ВЛС являются пользователь (преподаватель) и модуль обслуживания. Модуль обслуживания обладает единственным вариантом использования – это непосредственно запуск модуля проектирования ВЛС. Список вариантов использования, доступных пользователю: функции по работе с расширениями, функции по работе с моделью, функции по проектированию содержимого модели, которые включают в себя три большие подгруппы функций – функции по работе с объектами, переменными и поведением, – просмотр информации о модуле, закрытие модуля после окончания процесса проектирования.

Работа с расширениями (рисунок 11) подразумевает под собой подключение, просмотр информации о расширении (разработчик расширения, версия расширения, краткое описание расширения), отключение расширений.

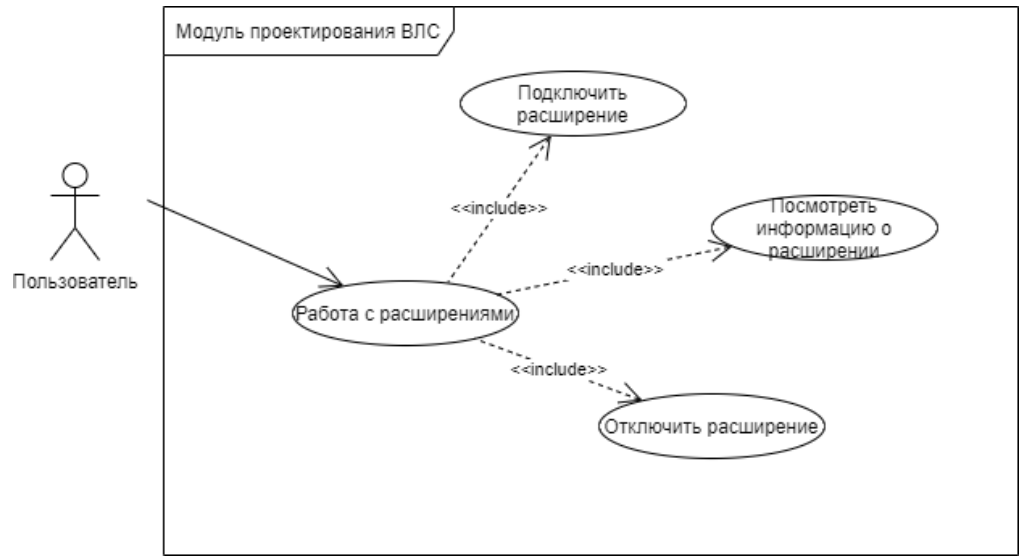


Рисунок 11 – Работа с расширениями

Работа с моделью (рисунок 12) включает в себя следующие варианты использования: создание, загрузку, изменение информации, сохранение модели.

Рисунок 12 – Работа с моделью

Работа с объектами (рисунок 13) включает в себя следующие варианты использования: группу функций по работе с полем проектирования, создание объекта модели, группу функций по работе с контейнером объекта, группу функций по работе с ассоциацией, удаление объекта модели.

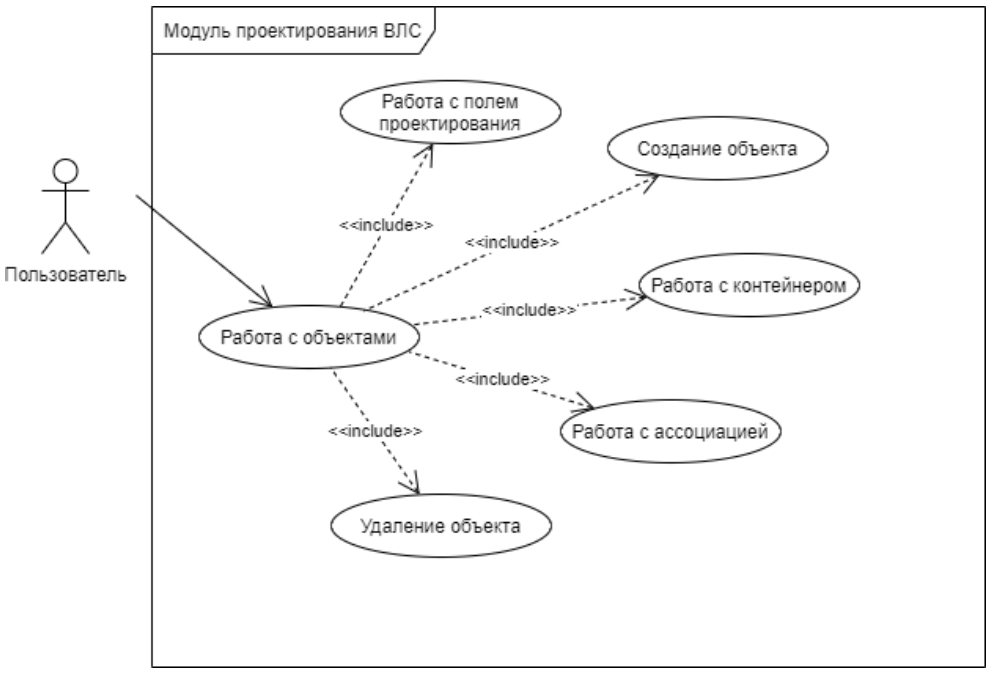


Рисунок 13 – Работа с объектами

Работа с полем проектирования (рисунок 14) состоит из следующих вариантов использования: изменение масштаба поля проектирования (есть три вариации масштаба: 50 процентов, 100 процентов, 150 процентов) и перемещение по полю проектирования с помощью полос прокрутки или с помощью курсора мыши через зажатие ее левой кнопки.

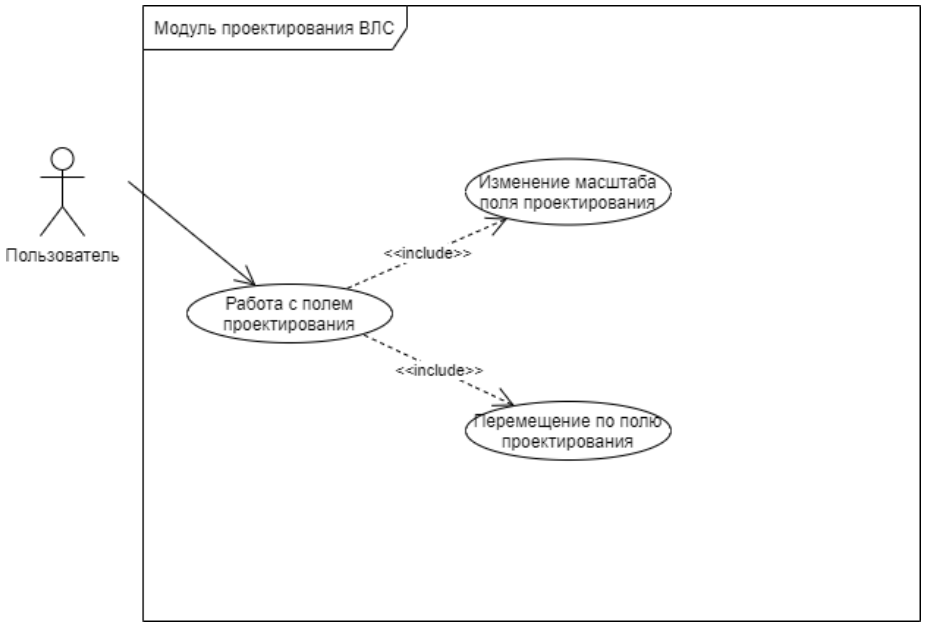


Рисунок 14 – Работа с полем проектирования

Работа с контейнером (рисунок 15) включает в себя следующие варианты использования: выделение контейнера, изменение размеров контейнера за счет вертикальных и горизонтальных якорей, перемещение

контейнера по полю проектирования за счет зажатия левой кнопки мыши при нахождении курсора над контейнером, снятие выделение с контейнера. Выделение контейнера нужно для совершения над ним различных манипуляций и просмотра списка переменных логики или интерфейса в зависимости от типа контейнера.

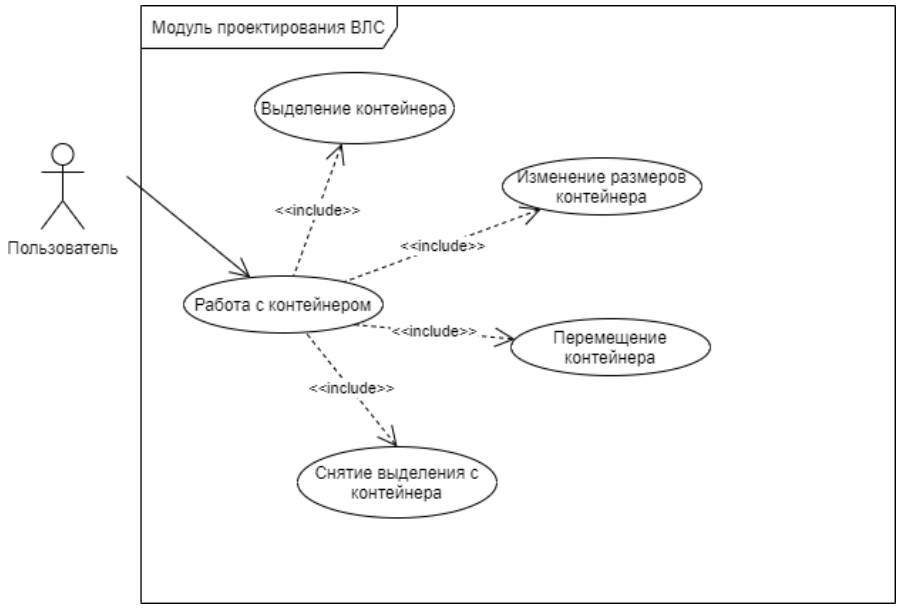


Рисунок 15 – Работа с контейнером

При работе с ассоциацией (рисунок 16) доступны следующие варианты использования: создание ассоциации между объектами, изменение ассоциации, то есть изменение формы линии, которая представляет собой графическое отображение ассоциации на поле проектирования логики, удаление ассоциации.

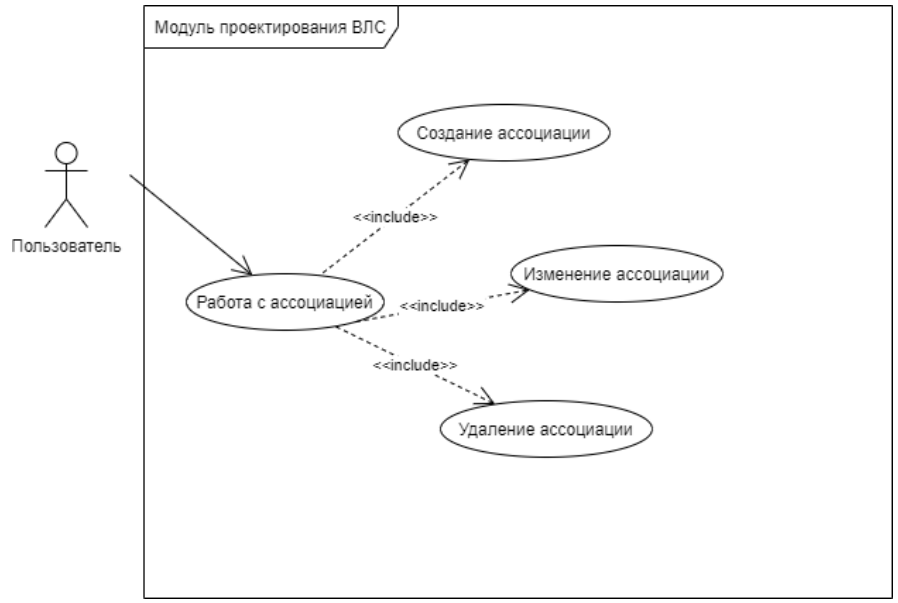


Рисунок 16 – Работа с ассоциацией

Работа с переменными (рисунок 17), состоит из следующих вариантов использования: получения списка переменных при выделении контейнера объекта, задания значения переменной при наличии такой возможности; создания пользовательской переменной логики; редактирование пользовательской переменной логики; удаления пользовательской переменной логики. Список переменных при получении отображается в виде таблицы свойств, и тип отображаемых переменных соответствует типу выделенного контейнера объекта. Для создания переменной необходимо указать название переменной, направление передачи данных, тип данных переменной и краткое описание переменной. Редактировать у переменной можно лишь название переменной и краткое описание.

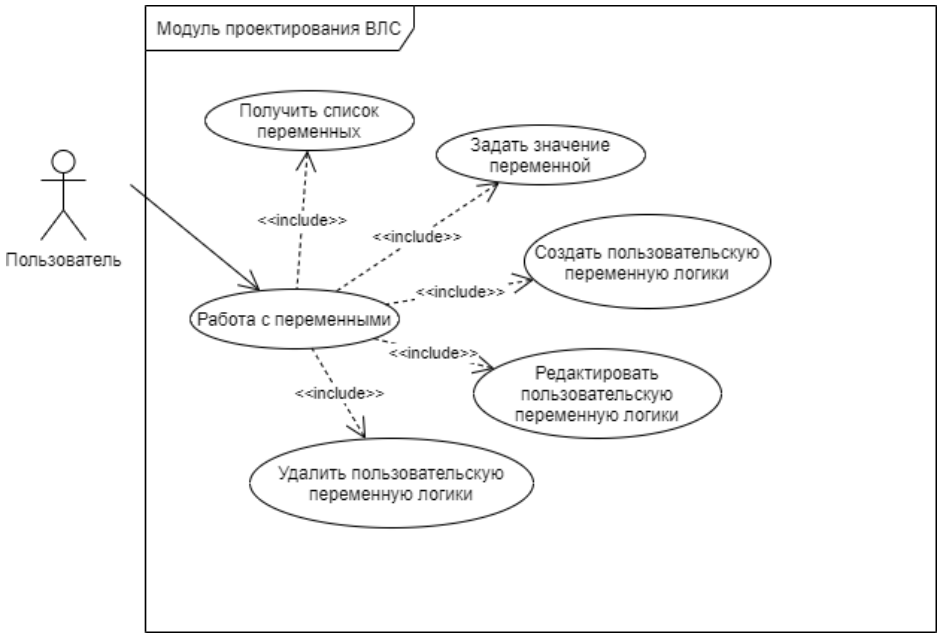


Рисунок 17 – Работа с переменными

Работа с поведением объекта (рисунок 18) включает в себя получение дерева поведения из структур данных в оперативной памяти для возможности просмотра и редактирования данного дерева пользователем, работу с конструкциями нижнего уровня языка LabScript, очищения поведения объекта от всех конструкций, сохранение дерева поведения в виде совокупности программ, каждая из которых связана с определенным событием.

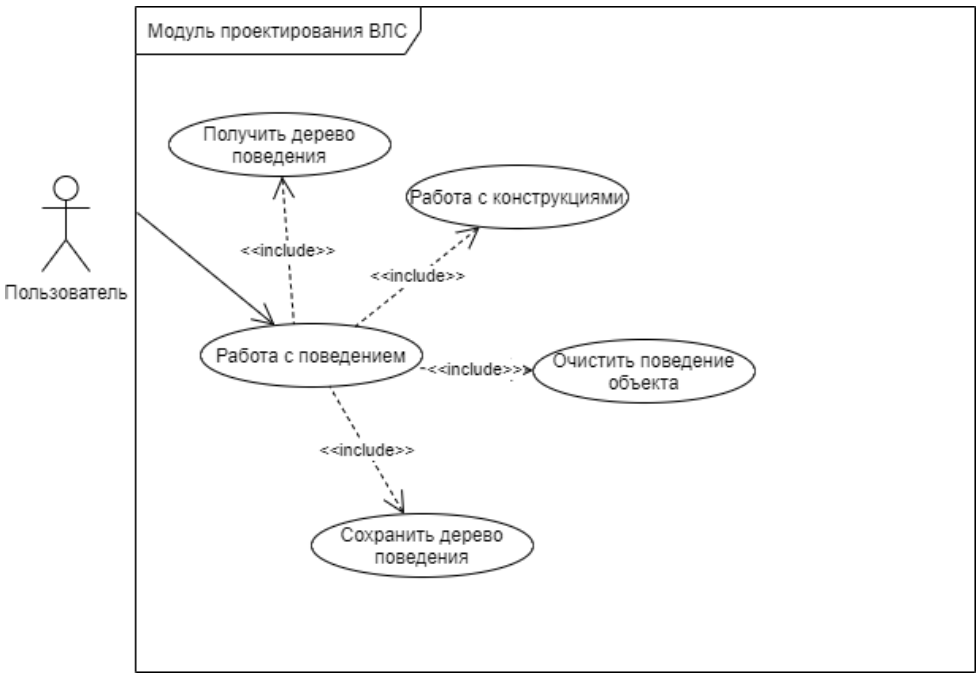


Рисунок 18 – Работа с поведением объекта

Работа с конструкциями (рисунок 19) состоит из создания конструкции, получения конструкции, группы функций по работе с операндами, изменения типа операции между операндами, группы функций по работе с выражениями, перемещения конструкции, удаления конструкции, сохранения конструкции. Получение конструкции представляет собой отображение конструкции в виде совокупности операндов и операций между ними для просмотра и редактирования пользователем. Перемещение конструкции заключается в изменении положения конструкции относительно соседних конструкций в рамках одного уровня вложенности. Сохранение конструкции представляет собой преобразование графического представления конструкции в виде совокупности операндов и операций в совокупность структур данных в оперативной памяти. Можно заметить, что группы функции по работе с операндами и по работе с выражениями связаны двумя отношениями. Это позволяет реализовать механизм произвольной вложенности конструкций и выражений.

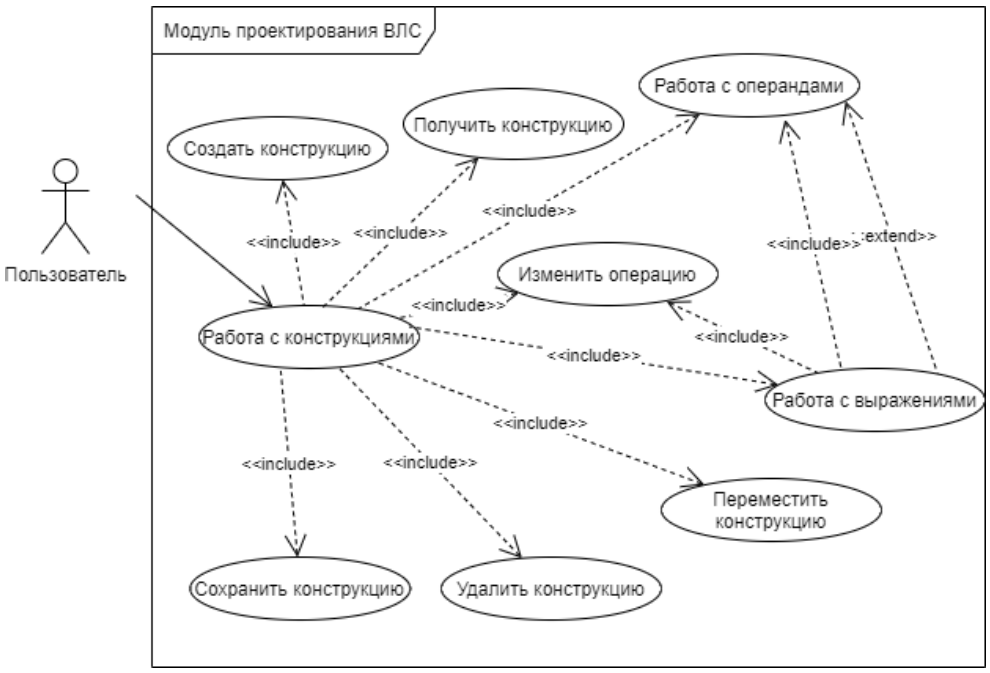


Рисунок 19 – Работа с конструкциями

Работа с операндами (рисунок 20) включает в себя добавление, изменение и удаление операнда. При добавлении операнда происходит добавление операции, за исключением случаев создания конструкции вызова и аргументов операнда вызова. При удалении операнда происходит удаление расположенной слева от него операции.

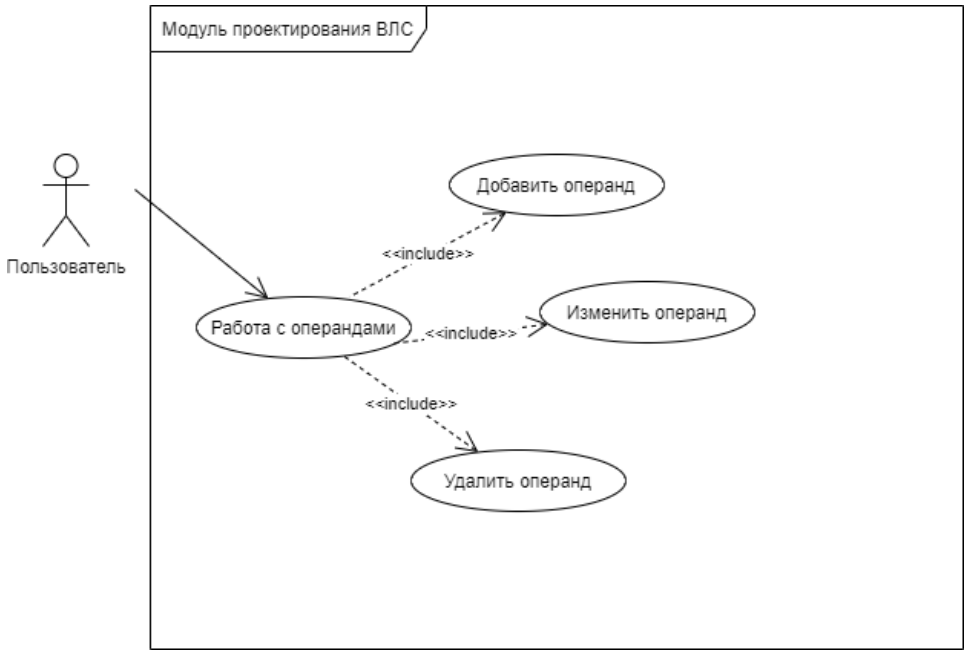


Рисунок 20 – Работа с операндами

Работа с выражениями (рисунок 21) включает получение выражения, группу функций по работе с операндами, изменение операции между операндами, сохранение выражения. На рисунке 21 отображены не все

перечисленные варианты использования, так как часть из них уже существует на рисунке 19. Получение и сохранение выражения аналогичны получению и сохранению конструкции.

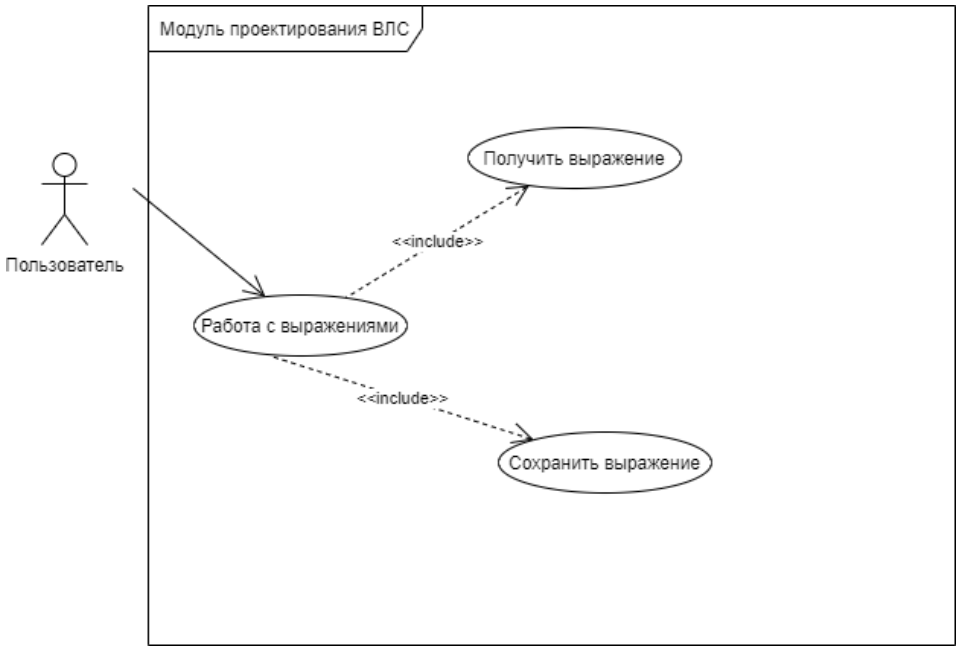


Рисунок 21 – Работа с выражениями

После идентификации набора функций, которые позволяет выполнять модуль проектирования ВЛС, необходимо рассмотреть подробнее часть из вышеперечисленных функций на основе диаграмм активности и деятельности. В качестве демонстрации результатов проектирования предлагаются к изучению функции: «Сохранить модель», «Загрузить модель», «Создать модель».

Функция «Сохранить модель» осуществляет процесс сериализации модели в файл с расширением «.lab». Вначале производится выбор месторасположения файла, далее – сохранение модели. При успешном выполнении обоих действий выводится сообщение об успехе. В противном случае выводятся сообщения об ошибках. Диаграммы активности и последовательности для данной функции представлены на рисунке Д.1 и в приложении Ж, лист 106 соответственно.

Функция «Загрузить модель» проверяет наличие текущей несохраненной модели, в случае наличия последней выводит соответствующее предупреждение. Если успешно пройден первый этап выполнения функции, то осуществляется выбор файла модели через диалог выбора, при успешном выборе файла, осуществляет процесс загрузки. Если процесс загрузки произведен успешно, то выводится сообщение об успехе. В остальных случаях выводятся сообщения об ошибках. Говоря о процессе загрузки, можно отметить, что это составной процесс, состоящий из двух

последовательных этапов: первичной загрузки, осуществляемой за счет стандартного механизма десериализации, и вторичной загрузки, осуществляемой за счет реализации разработанного интерфейса ILoadable. При первом этапе происходит создание объектов модели и заполнение их первичной информацией, которая содержит информацию о связях между структурами данных. На втором этапе происходит восстановление ссылочной целостности между структурами данных путем замены уникальных идентификаторов на ссылки на объекты классов в оперативной памяти. Диаграммы активности и последовательности для данной функции представлены на рисунке Д.2 и в приложении Ж, лист 107 соответственно.

Функция «Создать модель» осуществляет вывод окна ввода информации о ВЛС и при вводе корректных данных создает пустую модель, в противном случае выводит сообщение об ошибке. Диаграммы активности и последовательности для данной функции представлены на рисунке Д.3 и в приложении Ж, лист 108 соответственно.

Диаграмма состояний для данного модуля представлена на рисунке 22. Для того чтобы избежать путаницы в состояниях, было принято решение представить ее в общем виде, где состояние «{действие}» соответствует какому-либо состоянию выполнения действия из списка доступных пользователю функций на диаграммах вариантов использования. Вход в основной цикл работы программы осуществляется через инициализацию модуля; далее пользователь выполняет те или иные действия согласно процессу проектирования и получает обратную связь от программы в виде сообщений об успехе (частный случай сообщений об успехе – отсутствие сообщений) или сообщений об ошибке; выход осуществляется путем нажатия кнопки закрытия модуля.

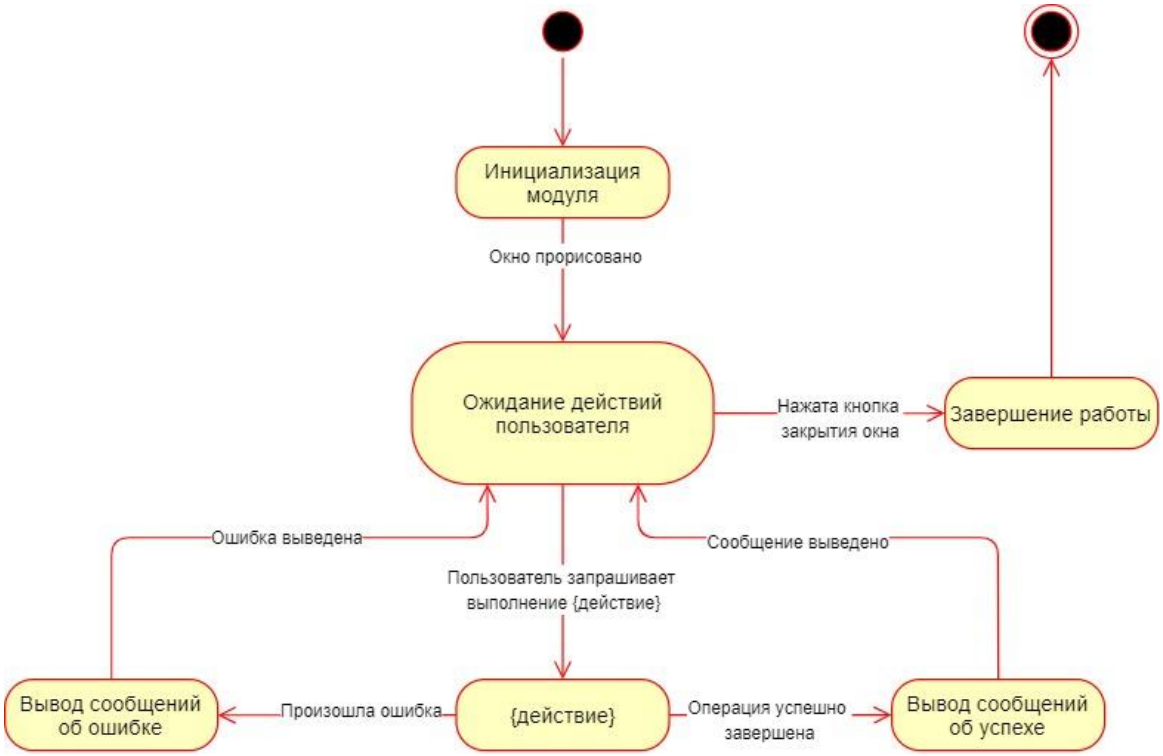


Рисунок 22 – Диаграмма состояний

Диаграммы классов модуля представлены в приложении И. Можно заметить, что все классы разбиты на две большие группы: классы интерфейса модуля и классы структур данных. Это сделано с той целью, что классы структур данных будут использоваться и в других модулях разрабатываемой САПР. Поэтому физически они также будут выполнены в виде двух динамических библиотек.

Рассматривая контекстную диаграмму классов интерфейса модуля, можно сказать, что основным классом в иерархии вложенности классов играет класс модуля проектирования. Он включает в себя ссылку на объект класса основного окна проектирования. Окно проектирования включает в себя менеджеры и связанные с ними классы.

Контекстная диаграмма компонентов модуля представлена на рисунке К.1.

Далее можно рассмотреть декомпозицию каждой библиотеки с точки зрения организации хранения исходного кода. Диаграммы декомпозиции для библиотеки модуля и библиотеки структур данных представлены на рисунках К.2 и К.3 соответственно. На диаграммах не отображены зависимости между компонентами для удобства восприятия ввиду большого числа компонентов на них.

Последняя диаграмма ввиду большого числа компонентов разбита лишь на каталоги. Декомпозиция каталогов представлена на рисунках К.4, К.5, К.6.

2.6 Выводы к разделу 2

В ходе выполнения процесса проектирования была проведена комплексная работа по формализации идей, положенных в основу модуля, во множество графических диаграмм, представляющих различные аспекты архитектуры модуля.

Особое место в проектировании заняло описание синтаксиса нижнего уровня реализации визуального языка программирования LabScript for Behaviour, который является фундаментальным средством для реализации логики работы модели в рамках данного модуля.

Также можно отметить разработку своего формата хранения данных модели, для чего были разработаны диаграммы сериализации модели.

3 Разработка модуля проектирования САПР LabCAD

После того, как этап проектирования пройден, необходимо приступать к программной реализации модуля с учетом выбранного языка программирования.

3.1 Выбор языка программирования

Важным моментом при разработке программного обеспечения является выбор инструмента реализации – языка программирования. Среди множества существующих в настоящий момент языков были выбраны для сравнения языки C, Java, Python, C++ и C# как одни из наиболее используемых в настоящее время [4].

В качестве критериев сравнения выбраны: опыт использования языка, порог вхождения, наличие средств для разработки графического пользовательского интерфейса, скорость работы программ в рамках решаемой задачи, поддержка объектно-ориентированного программирования. Под порогом вхождения подразумевается количество затраченных усилий на освоение языка на базовом уровне относительно других сравниваемых языков программирования.

Таблица результатов сравнения представлена ниже (таблица 3).

Таблица 3 – Сравнение языков программирования

Критерий сравнения	Язык программирования				
	C	Java	Python	C++	C#
1	2	3	4	5	6
Опыт использования языка	5 мес.	3 мес.	60 мес.	0 мес.	42 мес.
Порог вхождения	высокий	средний	низкий	высокий	средний
Средства для разработки графического пользовательского интерфейса	имеются	имеются	имеются	имеются	имеются

Продолжение таблицы 3

1	2	3	4	5	6
Скорость работы программ в рамках решаемой задачи	допустимая	допустимая	недопусти-мая	допустимая	допустимая
Поддержка объектно-ориентиро-ванного программи-рования	отсутствует	имеется	имеется	имеется	имеется

Можно сразу убрать из вариантов выбора языки программирования C и Python, так как первый не поддерживает объектно-ориентированное программирование [5], которое является основополагающим в данной работе, а второй имеет низкую скорость выполнения программ по сравнению с другими перечисленными языками [6], что неприемлемо для разрабатываемого модуля.

Из оставшихся языков следует выбрать язык программирования C#, что связано с тем, что необходимо разработать наукоемкий проект в короткие сроки одним разработчиком. В этих условиях встает необходимость использовать язык, на котором имеется опыт разработки, более того, язык программирования C# имеет значительные преимущества перед C++ в контексте скорости разработки; на языке Java опыт разработки незначительный, что может сказаться на качестве конечного продукта с учетом поставленных рамок.

3.2 Программная реализация модуля

В ходе кодирования была разработана программная реализация модуля проектирования виртуальных лабораторных стендов.

Весь интерфейс модуля можно разделить на несколько форм:

- форма главного окна;
- форма изменения информации о ВЛС;
- форма редактирования переменной;
- форма просмотра информации о расширении;
- форма информации о модуле;
- форма отключения расширений;
- форма редактирования поведения объекта;

- форма Мастера действий;
- форма информации об ошибках.

Следует остановиться подробнее на каждой форме.

Форма главного окна представляет собой основную форму модуля проектирования САПР, с которой работает пользователь в процессе проектирования. В общем виде она выглядит следующим образом (рисунок 23).

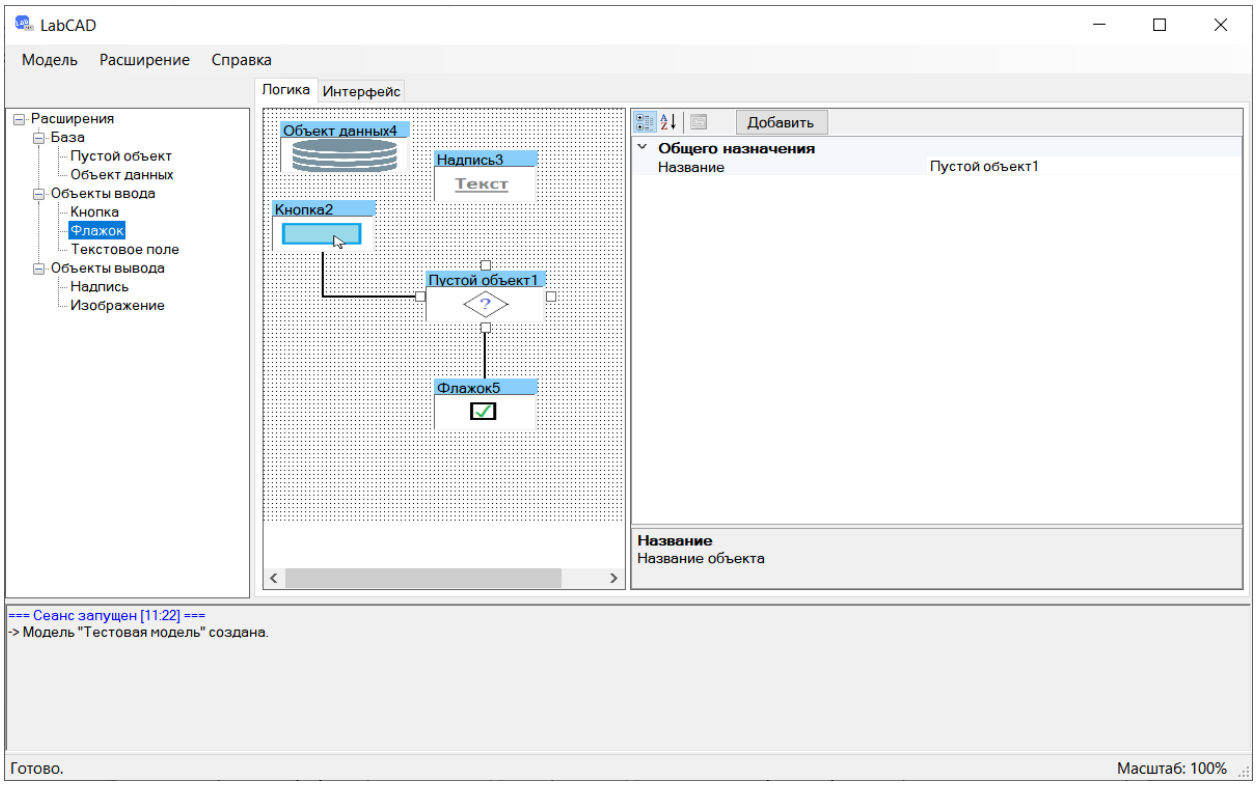


Рисунок 23 – Форма главного окна

Она включает в себя область главного меню, область строки состояния, область дерева расширений, область консоли и область проектирования. Область главного меню предоставляет основные действия по работе с моделью, расширениями и информацией о модуле. Область строки состояния выводит некоторую справочную информацию о состоянии модуля, например, масштаб текущего выбранного поля проектирования. Область дерева расширений (рисунок 24) необходима для отображения подключенных расширений, с которыми может работать пользователь. Область консоли (рисунок 25) предназначена для вывода некоторых выполняемых операций и результатов их выполнения. Область проектирования предназначена непосредственно для осуществления процесса проектирования. Она делится на область проектирования логики (рисунок 26) и область проектирования интерфейса (рисунок 27), каждая из

которых состоит из двух частей: поля проектирования и область менеджера переменных.

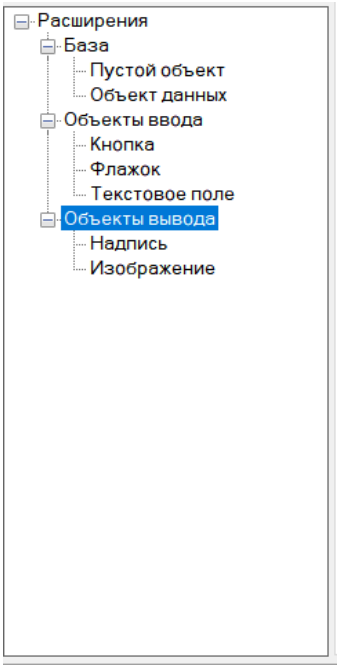


Рисунок 24 – Область дерева расширений

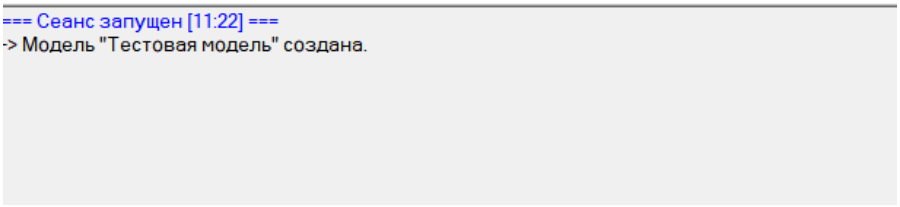


Рисунок 25 – Область консоли

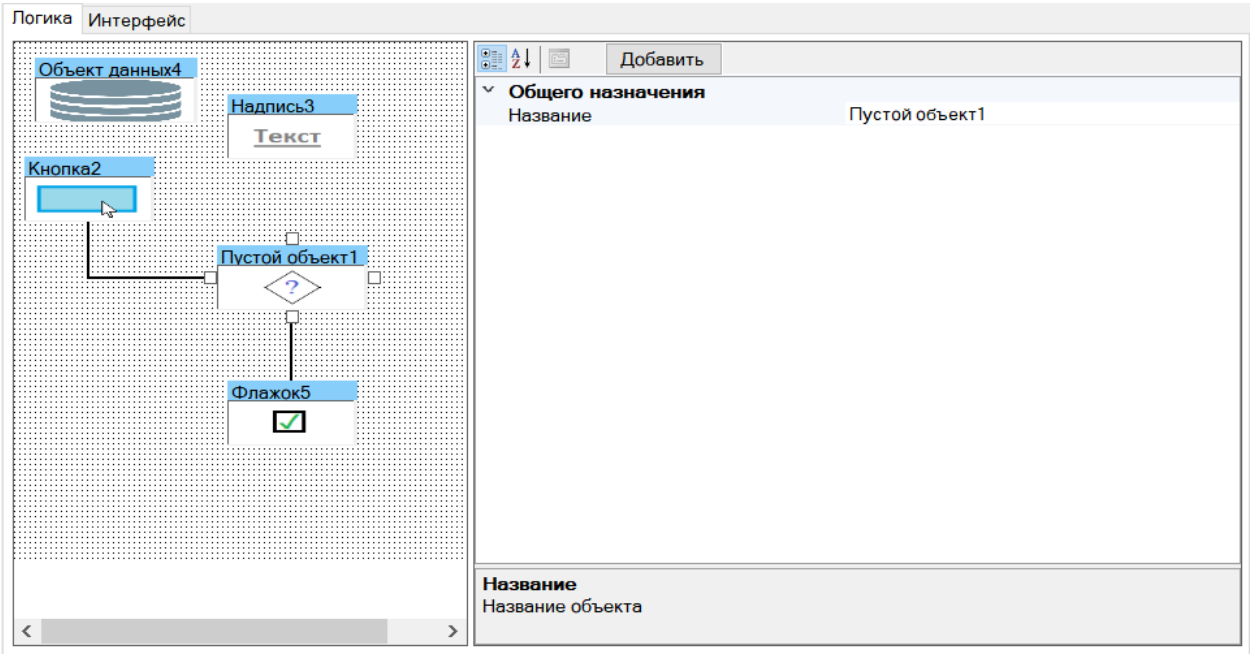


Рисунок 26 – Область проектирования логики

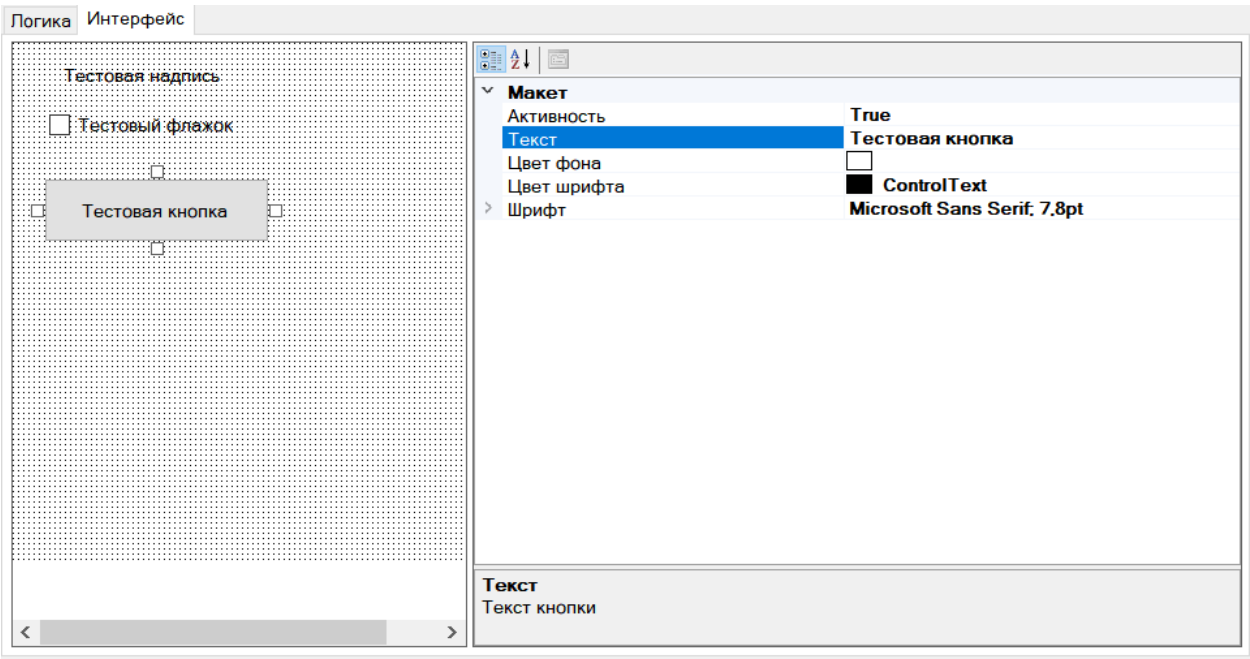


Рисунок 27 – Область проектирования интерфейса

Форма изменения информации о ВЛС (рисунок 28) предназначена как для создания новой модели, так и для редактирования информации о текущей модели.

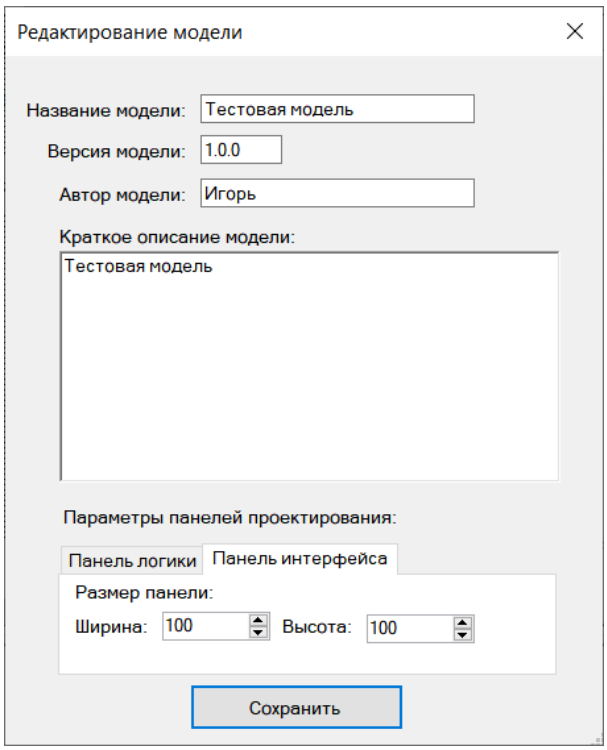


Рисунок 28 – Форма изменения информации о ВЛС

Форма редактирования переменной (рисунок 29) предназначена для создания переменной логики и изменения информации о имеющейся переменной логики.

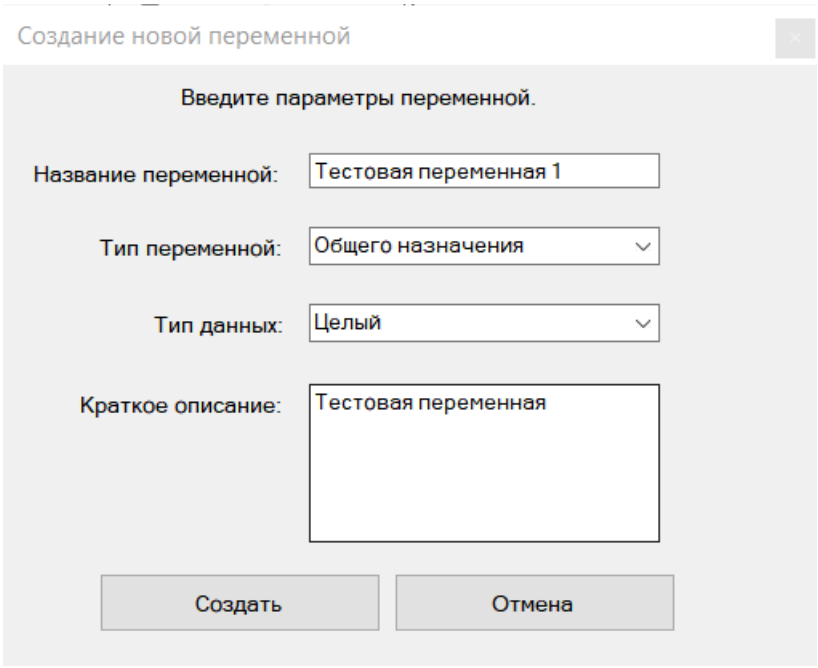


Рисунок 29 – Форма редактирования переменной

Форма просмотра информации о расширении (рисунок 30) позволяет увидеть краткую информацию о подключенном расширении.

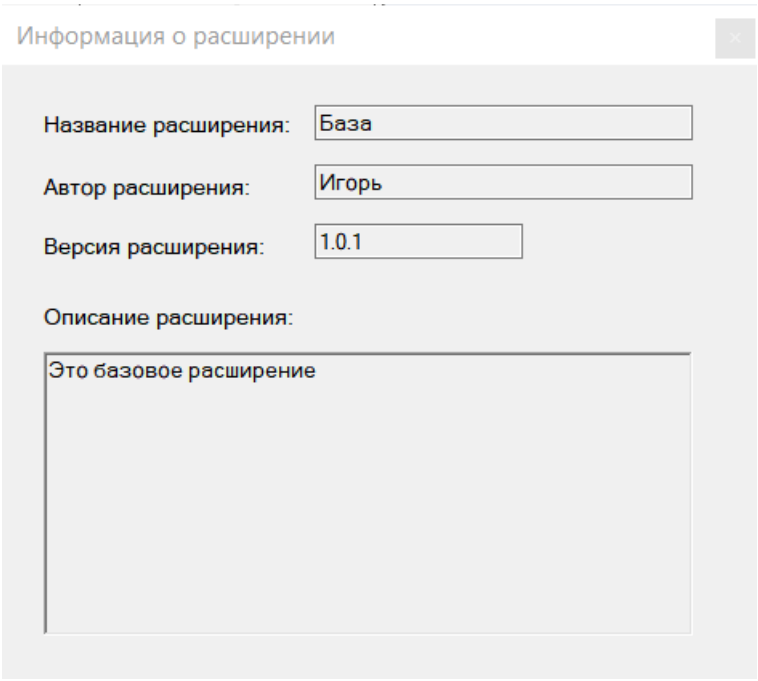


Рисунок 30 – Форма просмотра информации о расширении

Форма информации о модуле (рисунок 31) выводит краткую справочную информацию о модуле проектирования.

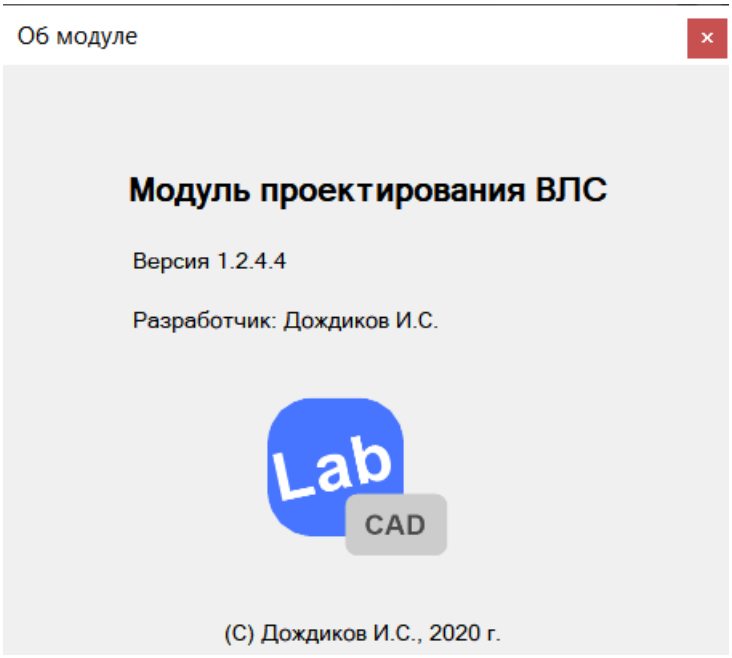


Рисунок 31 – Форма информации о модуле

Форма отключения расширений (рисунок 32) предназначена для отключения подключенных расширений путем выбора их из списка.

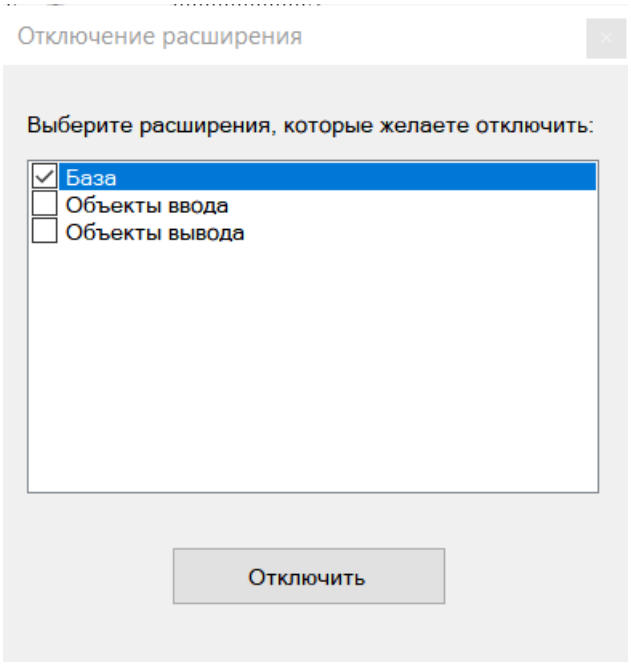


Рисунок 32 – Форма отключения расширений

Форма редактирования поведения объекта (рисунок 33) позволяет изменять поведение выбранного объекта.

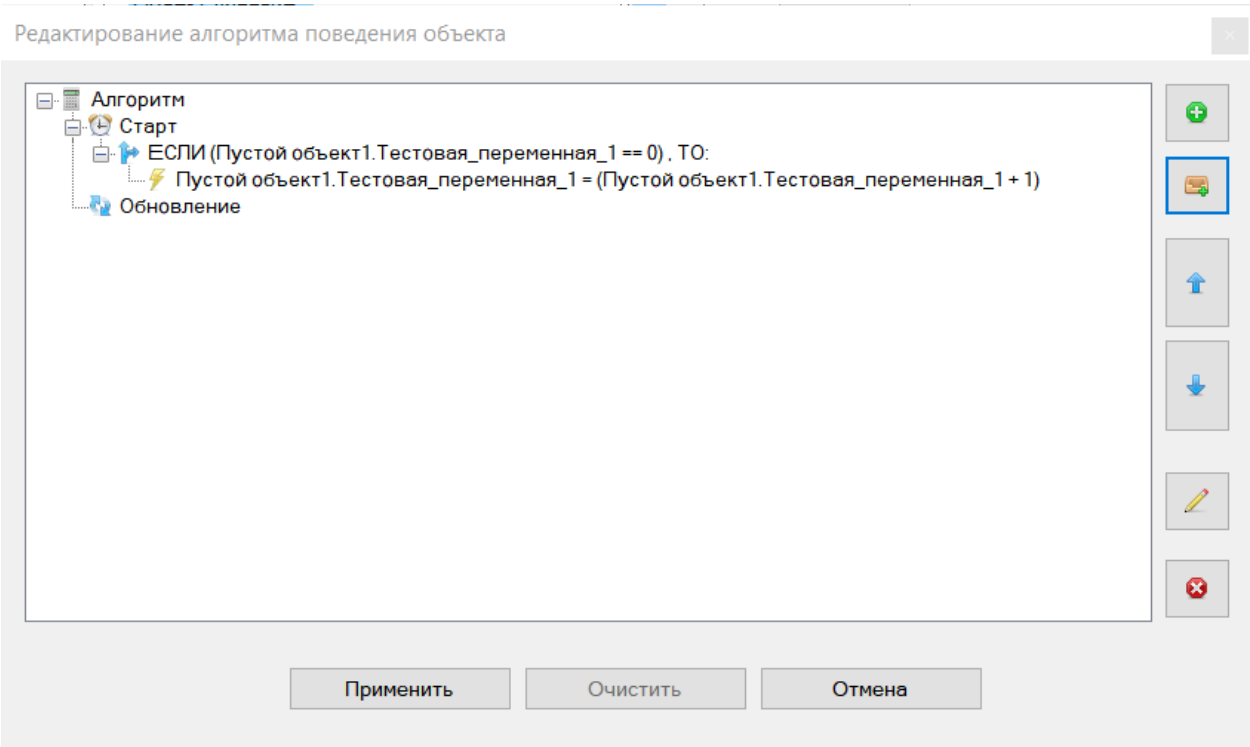


Рисунок 33 – Форма редактирования поведения объекта

Форма Мастера действий (рисунок 34) предназначена для создания и редактирования конструкций и выражений.

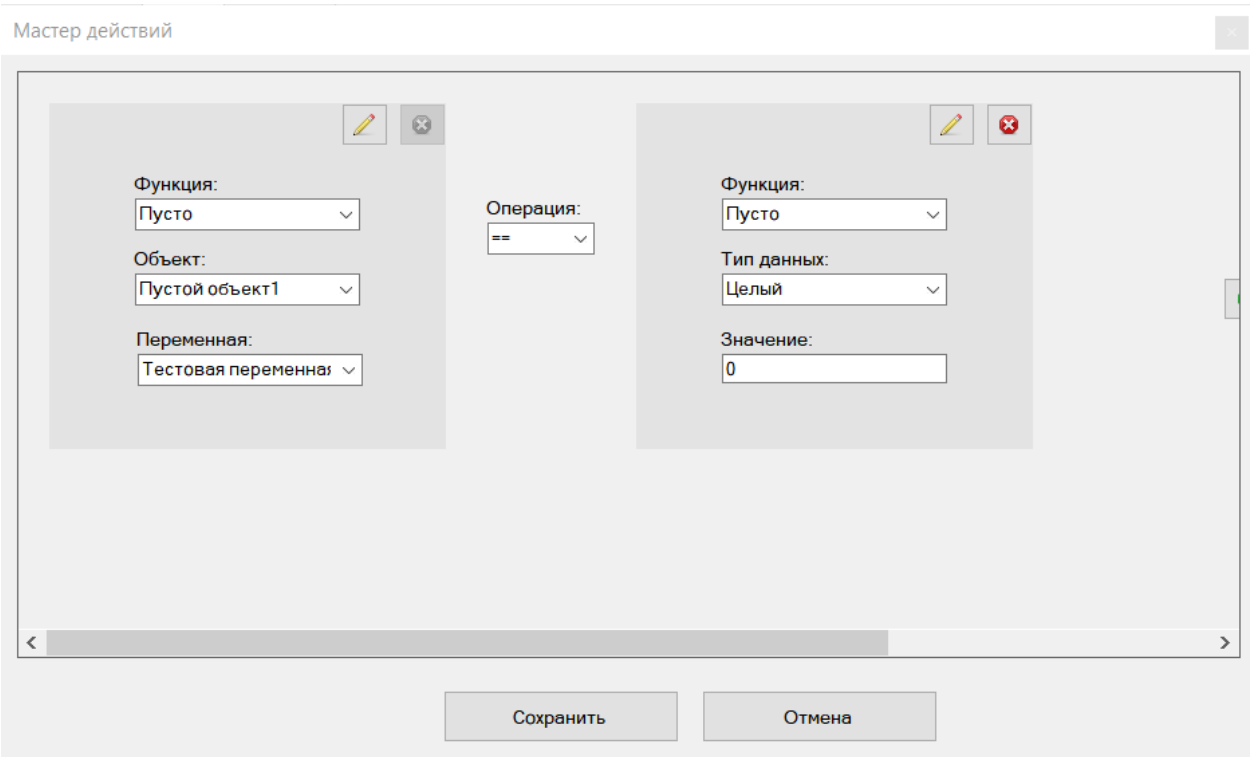


Рисунок 34 – Форма Мастера действий

Форма информации об ошибках (рисунок 35) позволяет увидеть список ошибок, возникших при создании или редактировании конструкции или выражения.

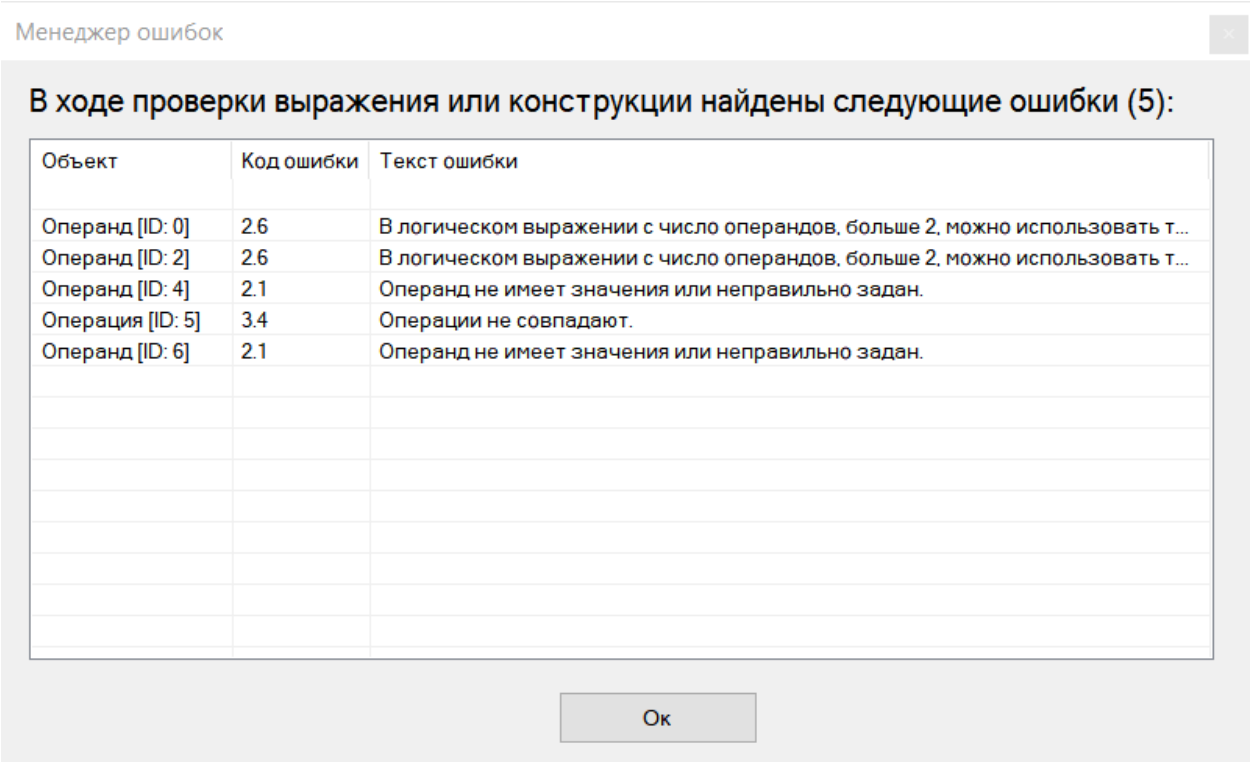


Рисунок 35 – Форма информации об ошибках

В качестве демонстрации результатов кодирования в приложении Л приведены фрагменты исходного кода модуля проектирования.

3.3 Разработка демонстрационного стенда «Модель температуры микроклимата теплицы»

В качестве демонстрации практической применимости разработанного модуля проектирования виртуальных лабораторных стендов была создана модель температуры микроклимата теплицы. Под температурой в контексте разработанного стенда следует понимать температуру воздуха.

Всего было выбрано два основных процесса для моделирования:

- изменение температуры окружающей среды;
- изменение температуры микроклимата теплицы.

С учетом того, что температура окружающей среды и температура микроклимата теплицы представляют собой физические величины, относительно медленно изменяющиеся в течение суток, были введены два вида виртуального времени: виртуальное время суток и вспомогательное виртуальное время, – для достижения необходимой для работы с моделью скорости изменения температур.

Виртуальное время среды измеряется в часах и минутах и предназначено для вывода в удобном для обучающегося виде и расчета скорости прироста температуры. Вспомогательное виртуальное время измеряется в часах и предназначено для расчета температуры окружающей среды.

Следует отметить, что наименования единиц времени в рамках текущей модели не отличаются от наименований единиц измерения реального времени, но эти единицы численно не равны друг другу. Минимальной единицей виртуального времени в контексте данной модели является минута, и она связана с тиками с помощью коэффициента ускорения, который принят за 5. То есть за один тик виртуальное время изменяется на 5 минут. Тик – это период времени между двумя последовательными во времени большими циклами работы модели.

Диапазон изменения первого вида виртуального времени соответствует диапазону изменения реального времени (24 часа). Диапазон измерения второго ограничен в 12 часов. Сущность ограничения вспомогательного виртуального времени в 12 часов заключается в том, что для получения полного времени суток переменная величины меняет знак изменения направления при достижении границ диапазона.

Температура микроклимата теплицы в данной модели зависит от температуры окружающей среды, коэффициента открытия форточек теплицы и виртуального времени среды. В свою очередь, температура окружающей среды является зависимой величиной от вспомогательного виртуального времени.

Для простоты модели были введены три состояния открытия форточек: закрыты, открыты наполовину, полностью открыты.

Далее следует рассмотреть математические модели выбранных процессов.

Для температуры окружающей среды эмпирическим путем была подобрана следующая формула:

$$T_{\text{окр}} = \frac{12.5}{1 + e^{-H_{\text{всп}}+8}} + 12.5, \quad (1)$$

где $T_{\text{окр}}$ – температура окружающей среды, °C ($T_{\text{окр}} \in [12.5,24.78]$);
$H_{\text{всп}}$ – вспомогательное виртуальное время, час. ($H_{\text{всп}} \in [0,12]$).

Согласно формуле (1), температура изменяется по сигмоидальному закону, что позволяет добиться плавности изменения температуры.

Для температуры микроклимата теплицы эмпирическим путем была подобрана следующая формула:

$$T_{\text{тепл}}^{1} = T_{\text{тепл}}^{0} + \Delta T_{\text{тепл}}, \quad (2)$$

где $T_{\text{тепл}}^{1}$ – температура микроклимата теплицы в момент времени (t+1) тик, °C;

$T_{\text{тепл}}^{0}$ – температура микроклимата теплицы в момент времени t тик, °C;

$\Delta T_{\text{тепл}}$ – скорость изменения температуры, °C/тик;

Скорость изменения температуры рассчитывается следующим образом:

$$\Delta T_{\text{тепл}} = \left(k_{\text{форточек}} + 0.02\right) * V_{\text{изм}} + \left(0.98 - k_{\text{форточек}}\right) * V_{\text{прироста}}, \quad (3)$$

где $k_{\text{форточек}}$ – коэффициент открытия форточек ($k_{\text{форточек}} \in [0,1]$);

$V_{\text{изм}}$ – скорость изменения температуры, °C/тик;

$V_{\text{прироста}}$ – скорость прироста температуры, °C/тик.

Согласно формуле (3), процесс изменения температуры микроклимата теплицы состоит из двух подпроцессов: подпроцесса установления равновесия температуры микроклимата теплицы с температурой окружающей среды при открытых форточках и подпроцесса прироста температуры микроклимата теплицы вследствие парникового эффекта.

Скорость изменения температуры рассчитывается по следующей формуле:

$$V_{\text{изм}} = \left(T_{\text{окр}} - T_{\text{тепл}}\right) * 0.5. \quad (4)$$

Скорость прироста температуры рассчитывается по следующей формуле:

$$V_{\text{прироста}} = \begin{cases} \left(0.5 - \dfrac{(H-7)^2}{98}\right) * 0.5, \text{если } H \in [2,14] \\ 0, \text{если } H \notin [2,14] \end{cases}, \quad (5)$$

где H – виртуальное время среды, час.

На основании данных формул был реализован демонстрационный стенд со следующим содержанием (рисунки 36, 37).

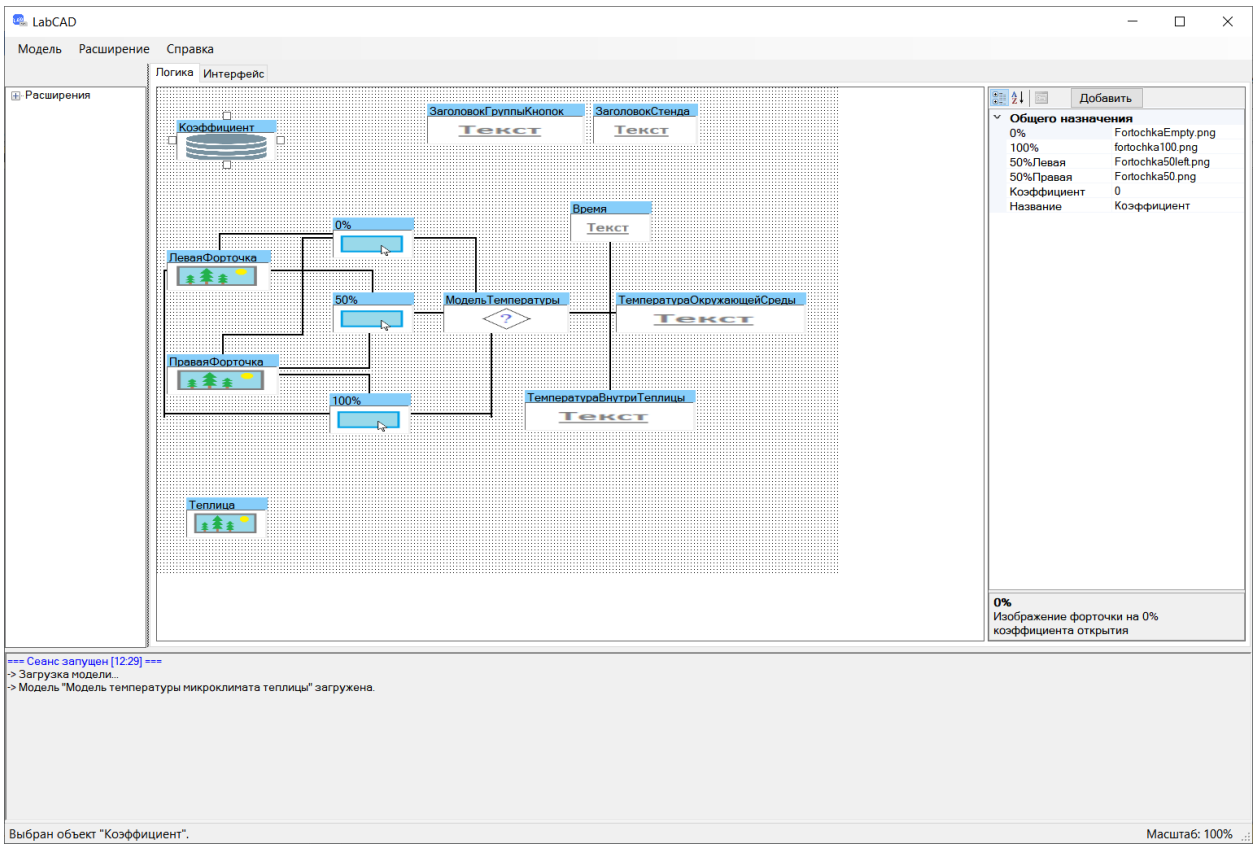


Рисунок 36 – Спроектированная логика демонстрационного стенда

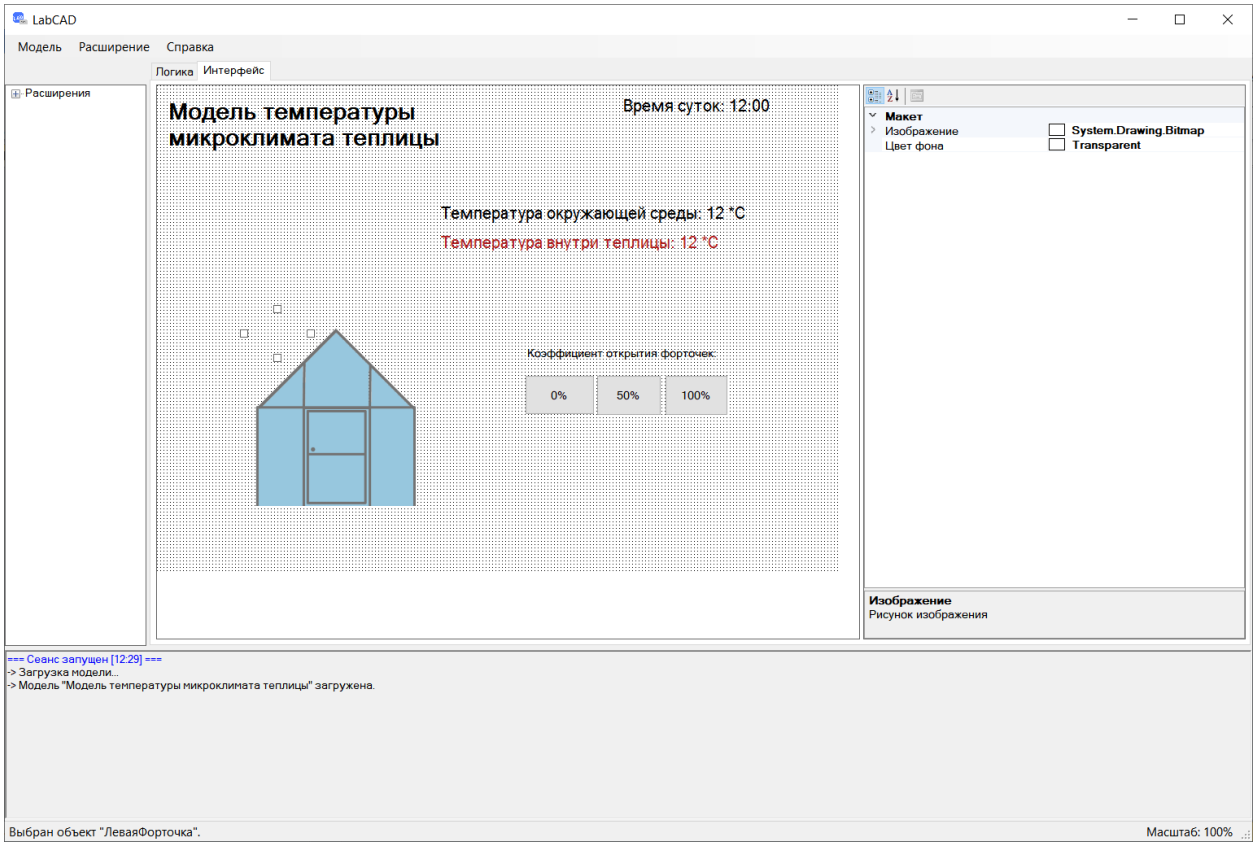


Рисунок 37 – Спроектированный интерфейс демонстрационного стенда

На рисунке 36 можно отметить объект «МодельТемпературы», который и содержит реализации математических моделей.

3.4 Выводы к разделу 3

В ходе программной реализации модуля был разработан модуль проектирования в виде динамической библиотеки САПР LabCAD с использованием языка программирования C#.

Весь интерфейс был разбит на несколько взаимосвязанных форм, центральной из которых является форма главного окна.

Разработан демонстрационный стенд температуры воздуха микроклимата теплицы.

При кодировании были использованы Интернет-ресурсы [7-8].

4 Технико-экономическое обоснование разработки модуля проектирования

Важное значение при разработке программного продукта имеет экономическое обоснование необходимости создания продукта, которое включает в себя расчет затрат на создание ПО, цены копии продукта, расчет выручки и прибыли от реализации заданного количества копий программы, расчет затрат, связанных с покупкой, внедрением и использованием ПО.

Следует заметить, что под фирмой в данном случае подразумевается группа, сформированная в рамках инженерного центра ВУЗа и состоящая из руководителя и инженера-программиста.

Целевая аудитория продукта – преподаватели образовательных учреждений основного общего и среднего (полного) общего образования, среднего профессионального образования, высшего образования.

Территория охвата продукта – Российская Федерация.

4.1 Расчет затрат на создание программного обеспечения, цены продукта

Профессионально-квалификационный состав группы разработчиков ПО представлен в таблице 4.

Таблица 4 – Профессионально-квалификационный состав группы разработчиков ПО

Наименование должности	Численность, чел.	Базовая ставка заработной платы, руб.	Повышающий коэффициент	Месячный оклад, руб.
Руководитель проекта	1	16000	1,5	24000
Инженер-программист	1	12500	1,5	18750

Для выполнения расчетов затрат и цены необходимо произвести определение трудоемкости выполнения работ по созданию ПО (таблица 5).

Таблица 5 – Трудоемкость выполнения работ

Наименование работы	Время выполнения, час.			Занятость команды, час.	
	Минимальное	Максимальное	Расчетное	Руководитель	Инженер-программист
1	2	3	4	5	6
Изучение задания	8	24	14.4	7.2	7.2
Подбор и изучение литературы	40	64	49.6	19.9	29.7
Анализ проблемы и существующих алгоритмов	40	64	49.6	19.9	29.7
Разработка общих принципов построения программы и методов представления данных	40	64	49.6	19.9	29.7
Финансово-экономическое обоснование создания ПО	20	32	24.8	24.8	0
Проведение маркетинговых исследований	20	32	24.8	24.8	0
Выбор системы управления данными, операционной системы, инструментария	16	40	25.6	12.8	12.8
Разработка структуры программного обеспечения	40	64	49.6	16	33.6
Разработка новых алгоритмов	40	64	49.6	8	41.6
Разработка пользовательского интерфейса	16	40	25.6	5	20.6
Написание программы	240	320	272	0	272

Продолжение таблицы 5

1	2	3	4	5	6
Отладка, тестирование, корректировка программы, устранение выявленных ошибок, выполнение	64	80	70.4	0	70.4
Разработка документации	24	40	30.4	0	30.4
Написание инструкции для пользователя на русском языке	8	16	11.2	5	6.2
Копирование ПО и подготовка его к реализации	40	64	49.6	40	9.6
Копирование документации и инструкции для пользователя	16	40	25.6	25.6	0
Всего	672	1048	822.4	228.9	593.5

Для удобства работы с данными таблицы 5, их необходимо сгруппировать по комплексам выполняемых работ (таблица 6).

Таблица 6 – Комплексы работ по созданию ПО

Наименование комплекса работ	Обозначение	Расчетное время выполнения, час.	Занятость команды, час.	
			Руководитель	Инженер-программист
1	2	3	4	5
1 Создание математического обеспечения и написания программы	$В_{мо}$	446.4	48.9	397.5
2 Отладка, тестирование, выполнение	$В_{м}$	70.4	0	30.4
3 Прочие затраты по разработке ПО	$В_{пр}$	164	84.6	79.4

Продолжение таблицы 6

1	2	3	4	5
4 Маркетинговые исследования	$В_{ми}$	24.8	24.8	0
5 Оформление программного продукта	$В_{оф}$	116.8	70.6	46.2
Всего	$В_{по}$	822.4	228.9	593.5

На основании таблицы 6 будет производиться расчет затрат на создание ПО.

Затраты на выплату зарплаты руководителям при создании математического обеспечения и написание программы рассчитываются по следующей формуле:

$$ЗП_{рук}^{мо} = \frac{Р_{рук} * В_{мо}^{рук} * О_{м}^{рук}}{Д_{с} * Д_{р}}, \quad (6)$$

где $Р_{рук}$ – число руководителей;

$В_{мо}^{рук}$ – время на создание математического обеспечения и написание программы, затраченное руководителем, час.;

$О_{м}^{рук}$ – месячный оклад руководителя, руб.;

$Д_{с}$ – длительность смены, час.;

$Д_{р}$ – среднее число рабочих дней в месяце, дней.

На основании формулы (6) можно получить следующее значение:

$$ЗП_{рук}^{мо} = \frac{Р_{рук} * В_{мо}^{рук} * О_{м}^{рук}}{Д_{с} * Д_{р}} = \frac{1 * 48.9 * 24000}{8 * 21} = 6985.71 \text{ руб.}$$

Затраты на выплату зарплаты программистам при создании математического обеспечения и написание программы рассчитываются по следующей формуле:

$$ЗП_{п}^{мо} = \frac{Р_{п} * В_{мо}^{п} * О_{м}^{п}}{Д_{с} * Д_{р}}, \quad (7)$$

где $Р_{п}$ – число программистов;

$В_{мо}^{п}$ – время на создание математического обеспечения и написание программы, затраченное программистом, час.;

$О_{м}^{п}$ – месячный оклад программиста, руб.;

$Д_{с}$ – длительность смены, час.;

$Д_{р}$ – среднее число рабочих дней в месяце, дней.

На основании формулы (7) можно получить следующее значение:

$$ЗП_{п}^{мо} = \frac{Р_{п} * В_{мо}^{п} * О_{м}^{п}}{Д_{с} * Д_{р}} = \frac{1 * 397.5 * 18750}{8 * 21} = 44363.84 \text{ руб.}$$

Затраты на заработную плату работников, участвующих в создании математического обеспечения и написании программы, рассчитываются по следующей формуле:

$$ЗП_{мо} = ЗП_{рук}^{мо} + ЗП_{п}^{мо}, \quad (8)$$

где $ЗП_{рук}^{мо}$ – затраты на выплату зарплаты руководителям при создании математического обеспечения и написания программы, руб.;
$ЗП_{п}^{мо}$ – затраты на выплату зарплаты программистам при создании математического обеспечения и написания программы, руб.

На основании формулы (8) можно получить следующее значение:

$$ЗП_{мо} = ЗП_{рук}^{мо} + ЗП_{п}^{мо} = 6985.71 + 44363.84 = 51349.55 \text{ руб.}$$

Фонд оплаты труда работников, участвующих в создании математического обеспечения и написания программы, рассчитывается по следующей формуле:

$$ФОТ_{мо} = ЗП_{мо} + П + В_{рк}, \quad (9)$$

где $ЗП_{мо}$ – затраты на выплату зарплаты работникам при создании математического обеспечения и написания программы, руб.;
П – премия, предусмотренная для работников, участвующих в создании ПО, руб. (20% от $ЗП_{мо}$);
$В_{рк}$ – выплаты по районному коэффициенту (установлены для г. Кирова в размере 15% от ($ЗП_{мо}$ + П)).

На основании формулы (9) можно получить следующее значение:

$$ФОТ_{мо} = 51349.55 + 10269.91 + 9242.92 = 70862.38 \text{ руб.}$$

Затраты на выплату зарплаты руководителям при создании ПО рассчитываются по следующей формуле:

$$ЗП_{рук}^{по} = \frac{Р_{рук} * В_{по}^{рук} * О_{м}^{рук}}{Д_{с} * Д_{р}}, \quad (10)$$

где $Р_{рук}$ – число руководителей;
$В_{по}^{рук}$ – время на создание ПО, затраченное руководителем, час.;
$О_{м}^{рук}$ – месячный оклад руководителя, руб.;
$Д_{с}$ – длительность смены, час.;
$Д_{р}$ – среднее число рабочих дней в месяце, дней.

На основании формулы (10) можно получить следующее значение:

$$ЗП_{рук}^{по} = \frac{Р_{рук} * В_{по}^{рук} * О_{м}^{рук}}{Д_{с} * Д_{р}} = \frac{1 * 228.9 * 24000}{8 * 21} = 32700 \text{ руб.}$$

Затраты на выплату зарплаты программистам при создании ПО рассчитываются по следующей формуле:

$$ЗП_{п}^{по} = \frac{Р_{п} * В_{по}^{п} * О_{м}^{п}}{Д_{с} * Д_{р}}, \quad (11)$$

где $Р_{п}$ – число программистов;

$В_{по}^{п}$ – время на создание ПО, затраченное программистом, час.;

$О_{м}^{п}$ – месячный оклад программиста, руб.;

$Д_{с}$ – длительность смены, час.;

$Д_{р}$ – среднее число рабочих дней в месяце, дней.

На основании формулы (11) можно получить следующее значение:

$$ЗП_{п}^{по} = \frac{Р_{п} * В_{по}^{п} * О_{м}^{п}}{Д_{с} * Д_{р}} = \frac{1 * 593.5 * 18750}{8 * 21} = 66238.84 \text{ руб.}$$

Затраты на заработную плату работников, участвующих в создании ПО, рассчитываются по следующей формуле:

$$ЗП_{об} = ЗП_{рук}^{по} + ЗП_{п}^{по}, \quad (12)$$

где $ЗП_{рук}^{по}$ – затраты на выплату зарплаты руководителям при создании ПО, руб.;

$ЗП_{п}^{по}$ – затраты на выплату зарплаты программистам при создании ПО, руб.

На основании формулы (12) можно получить следующее значение:

$$ЗП_{об} = ЗП_{рук}^{по} + ЗП_{п}^{по} = 32700 + 66238.84 = 98938.84 \text{ руб.}$$

Общий фонд оплаты труда работников, участвующих в создании ПО, рассчитывается по следующей формуле:

$$ФОТ_{об} = ЗП_{об} + П + В_{рк}, \quad (13)$$

где $ЗП_{мо}$ – затраты на выплату зарплаты работникам при создании математического обеспечения и написания программы, руб.;

П – премия, предусмотренная для работников, участвующих в создании ПО, руб. (20% от $ЗП_{об}$);

$В_{рк}$ – выплаты по районному коэффициенту (установлены для г. Кирова в размере 15% от ($ЗП_{об}$ + П)).

На основании формулы (13) можно получить следующее значение:

$$ФОТ_{об} = 98938.84 + 19787.77 + 17808.99 = 136535.60 \text{ руб.}$$

Затраты на создание математического обеспечения и написание программы, рассчитываются по следующей формуле:

$$З_{мо} = ЗП_{мо} + П + В_{рк} + С_{сн} * ФОТ_{мо} + Н_{р}, \quad (14)$$

где $ЗП_{мо}$ – Затраты на заработную плату работников, участвующих в создании математического обеспечения и написании программы, руб.;

П – премия, предусмотренная для работников, участвующих в создании ПО, руб. (20% от $ЗП_{мо}$);

$В_{рк}$ – выплаты по районному коэффициенту (установлены для г. Кирова в размере 15% от ($ЗП_{мо}$ + П));

$С_{сн}$ – общая ставка страховых взносов (30% и ФСС СН 0.2%);

$ФОТ_{мо}$ – фонд оплаты труда работников, участвующих в создании математического обеспечения и написания программы, руб.;

$Н_{р}$ - накладные расходы организации, где разрабатывается ПО (затраты на отопление, освещение, на содержание административно-управленческого персонала и др.) (120% от $ЗП_{мо}$).

На основании формулы (14) можно получить следующее значение:

$$З_{мо} = ЗП_{мо} + П + В_{рк} + С_{сн} * ФОТ_{мо} + Н_{р} = 51349.55 +$$

$$10269.9 + 9242.92 + 0.32 * 70862.38 + 61619.46 = 155157.79 \text{ руб.}$$

Стоимость одного часа эксплуатации компьютера рассчитывается по следующей формуле:

$$С_{м} = \frac{Т_{с}^{1} * Т_{к} * К_{нр}}{Д_{с} * Д_{р}}, \quad (15)$$

где $Т_{с}^{1}$ – минимальная заработная плата, законодательно установленный размер на текущий момент, руб.;

$Т_{к}$ – повышающий коэффициент, соответствующий ставке инженера-программиста;

$К_{нр}$ – коэффициент, учитывающий накладные и другие расходы, связанные с работой компьютера (можно принять равным 3% от $Т_{с}^{1}$);

$Д_{с}$ – длительность смены, час.;

$Д_{р}$ – среднее число рабочих дней в месяце, дней.

На основании формулы (15) можно получить следующее значение:

$$С_{м} = \frac{Т_{с}^{1} * Т_{к} * К_{нр}}{Д_{с} * Д_{р}} = \frac{12130 * 1.5 * 1.03}{8 * 21} = 111.55 \text{ руб.}$$

Затраты, связанные с работой компьютера при разработке ПО, рассчитываются по следующей формуле:

$$З_{ком} = (В_м + В_{нп}) * С_м, \quad (16)$$

где $В_м$ – время, затраченное на отладку, тестирование, выполнение программы, час.;

$В_{нп}$ – время, затраченное на написание программы, час.;

$С_м$ – стоимость одного часа эксплуатации компьютера, руб.

На основании формулы (16) можно получить следующее значение:

$$З_{ком} = (В_м + В_{нп}) * С_м = (70.4 + 272) * 111.55 = 38194.72 \text{ руб.}$$

Затраты на разработку ПО рассчитываются по следующей формуле:

$$З_{рп} = З_{мо} + З_{ком} + З_{пр}, \quad (17)$$

где $З_{мо}$ – затраты на создание математического обеспечения и написание программы, руб.;

$З_{ком}$ – затраты, связанные с работой компьютера при разработке ПО, руб.;

$З_{пр}$ – прочие затраты, связанные с разработкой ПО (изучение задания, литературы, патентов, анализ проблемы и существующих алгоритмов, проведение экономических расчетов и др.), их можно принять в размере 20% от $З_{мо}$, руб.

На основании формулы (17) можно получить следующее значение:

$$З_{рп} = З_{мо} + З_{ком} + З_{пр} = 155157.79 + 38194.72 + 31031.56 =$$

$$= 224384.07 \text{ руб.}$$

Результатом расчетов является смета затрат на создание ПО (таблица 7).

Таблица 7 – Смета затрат на создание ПО

Наименование статьи затрат	Буквенное обозначение	Сумма, руб.
1	2	3
Зарплата программистов	$ЗП_п^{мо}$	44363.84
Зарплата руководителей	$ЗП_{рук}^{мо}$	6985.71
Итого зарплаты, израсходованной на создание математического обеспечения и написание программы	$ЗП_{мо}$	51349.55
Премия	П	10269.90
Выплаты по районному коэффициенту	$В_{рк}$	9242.92
Страховые взносы с ФОТ	$С_{сн}$	22675.96
Накладные расходы	$Н_р$	61619.46
Итого затрат на создание ПО и написание программы	$З_{мо}$	155157.79

Продолжение таблицы 7

1	2	3
Затраты, связанные с работой компьютера при разработке ПО	$З_{ком}$	38194.72
Прочие затраты, связанные с разработкой ПО	$З_{пр}$	31031.56
Итого затрат на разработку ПО	$З_{рп}$	224384.07
Налоги, включаемые в затраты на создание программы	$Н_{сп}$	20480.34
Затраты на оформление программного продукта	$З_{оф}$	33657.61
Затраты на маркетинговые исследования	$З_{ми}$	22438.41
Всего затрат на создание ПО	$З_{сп}$	300960.43

Следует заметить, что налоги, включаемые в затраты на создание программы, составляют 15% от общего фонда оплаты работников, затраты на оформление программного продукта и на маркетинговые исследования составляют 15% и 10% от общих затрат на разработку ПО соответственно.

Далее необходимо рассчитать проектные цены.

При продаже одной копии ПО получить значения цен создания и реализации можно следующим образом.

Величина прибыли рассчитывается по следующей формуле:

$$Пр = З_{сп} * У_{р}, \quad (18)$$

где $З_{сп}$ – общие затраты на разработку ПО, руб.;
$У_{р}$ – уровень рентабельности программного продукта (20%).

На основании формулы (18) можно получить следующее значение:

$$Пр = З_{сп} * У_{р} = 300960.43 * 0.2 = 60192.09 \text{ руб.}$$

Цена создания рассчитывается по следующей формуле:

$$Ц_{с} = З_{сп} + Пр, \quad (19)$$

где $З_{сп}$ – общие затраты на разработку ПО, руб.;
Пр – величина прибыли, руб.

На основании формулы (19) можно получить следующее значение:

$$Ц_{с} = З_{сп} + Пр = 300960.43 + 60192.09 = 361152.52 \text{ руб.}$$

Розничная цена ПО рассчитывается по следующей формуле:

$$Ц_{р} = Ц_{с} + НДС + Тн, \quad (20)$$

где $Ц_{с}$ – цена создания, руб.;
НДС – налог на добавленную стоимость (20%);

Тн – торговая наценка при реализации ПО через специализированные магазины (торговых посредников) (10%).

На основании формулы (20) можно получить следующее значение:

$$Ц_р = Ц_с + НДС + Тн = 361152.52 + 72230.50 + 36115.25 =$$

$$= 469498.27 \text{ руб.}$$

При реализации нескольких копий ПО формулы расчета цен отличаются.

Затраты на одно копирование рассчитывается по следующей формуле:

$$З_{коп} = \frac{(В_{коп} + В_{под}) * С_м}{60} + Ц_н + З_{док}, \quad (21)$$

где $В_{коп}$ – время одного копирования ПО, мин.;

$В_{под}$ – время подготовки (форматирования) носителя информации, мин.;

$С_м$ – стоимость одного часа эксплуатации компьютера, руб.;

$Ц_н$ – розничная цена носителя информации, используемого под копию ПО, руб.;

$З_{док}$ – затраты на копирование или печатание сопроводительной документации (инструкции для пользователя и др.) и приобретение упаковки для хранения этой документации и носителя информации (20% от минимальной заработной платы), руб.

На основании формулы (21) можно получить следующее значение:

$$З_{коп} = \frac{(В_{коп} + В_{под}) * С_м}{60} + Ц_н + З_{док} = \frac{(2+4) * 111.55}{60} + 65 +$$

$$+2426 = 2502.16 \text{ руб.}$$

Планируется рассмотреть цены для 5, 10, 20, 50, 100, 150 копий ПО.

Цена создания для нескольких копий рассчитывается по следующей формуле:

$$Ц_с = \left(\frac{З_{сп}}{N_{коп}} + З_{коп}\right) * (1 + У_р), \quad (22)$$

где $З_{сп}$ – общие затраты на разработку ПО, руб.;

$N_{коп}$ – количество копий, снимаемых с оригинала ПО, шт.;

$З_{коп}$ – затраты на одно копирование, руб.;

$У_р$ – уровень рентабельности программного продукта (20%).

Розничная цена ПО для нескольких копий рассчитывается по следующей формуле

$$Ц_р = Ц_с * (1 + НДС) * (1 + Тн), \quad (23)$$

где $Ц_с$ – цена создания, руб.;
НДС – налог на добавленную стоимость (20%);
Тн – торговая наценка при реализации ПО через специализированные магазины (торговых посредников) (10%).

Для примера можно рассчитать цену создания и цену реализации ПО для пяти копий.

На основании формулы (22) можно получить следующее значение цены создания для пяти копий:

$$Ц_с = \left(\frac{З_{сп}}{N_{коп}} + З_{коп}\right) * \left(1 + У_р\right) = \left(\frac{300960.43}{5} + 2502.16\right) *$$

$$* (1 + 0.2) = 75233.10 \text{ руб.}$$

На основании формулы (23) можно получить следующее значение розничной цены ПО для пяти копий:

$$Ц_р = Ц_с * (1 + НДС) * (1 + Тн) = 75233.10 * (1 + 0.2) *$$

$$* (1 + 0.1) = 99307.69 \text{ руб.}$$

По аналогии можно рассчитать цены создания и розничные цены ПО для другого количества копий. В итоге получается следующая таблица (таблица 8).

Таблица 8 – Зависимость между минимальной ценой и числом реализуемых копий ПО

Число реализуемых копий	Цена создания, руб.	Розничная цена, руб.
1	361152.52	469498.27
5	75233.10	99307.69
10	39117.84	51635.55
20	21060.22	27799.49
50	10225.64	13497.84
100	6614.12	8730.64
150	5410.28	7141.57

Планируется продать 100 копий, соответственно, итоговая цена ПО будет составлять 8730.64 рублей. Решение основано на том, что продукт

будет продаваться на территории Российской Федерации, в результате, за счет охвата значительного количества образовательных организаций, величина спроса будет иметь немалое значение.

4.2 Расчет выручки и прибыли от реализации программного продукта

После расчета затрат и цен нужно определить предполагаемую сумму выручки и прибыли от продажи программного продукта.

Валовая выручка от реализации ПО по рыночной цене (без учета торговой наценки) рассчитывается по следующей формуле:

$$ВР_{бр} = Ц'_р * N_{коп}, \quad (24)$$

где $Ц'_р$ – цена реализации разработанного ПО (без учета наценки), руб.;

$N_{коп}$ – количество копий, снимаемых с оригинала ПО, шт..

На основании формулы (24) можно получить следующее значение:

$$ВР_{бр} = Ц'_р * N_{коп} = 7936.94 * 100 = 793694 \text{ руб.}$$

Объем выручки от продажи ПО по цене создания рассчитывается по следующей формуле:

$$ВР_н = Ц_с * N_{коп}, \quad (25)$$

где $Ц_с$ – цена создания, руб.;

$N_{коп}$ – количество копий, снимаемых с оригинала ПО, шт.

На основании формулы (25) можно получить следующее значение:

$$ВР_н = Ц_с * N_{коп} = 6614.12 * 100 = 661412 \text{ руб.}$$

Величина прибыли от продажи всех копий программного продукта рассчитывается по следующей формуле:

$$П_{валов} = ВР_н - З_{об}, \quad (26)$$

где $ВР_н$ – объем выручки от продажи ПО по цене создания, руб.;

$З_{об}$ – общие затраты на создание и копирование всех реализуемых ПО, руб.

На основании формулы (26) можно получить следующее значение:

$$П_{валов} = ВР_н - З_{об} = 661412 - 551176.43 = 110235.57 \text{ руб.}$$

Балансовая прибыль, которую может получить фирма (организация), разрабатывающая и реализующая ПО, рассчитывается по следующей формуле:

$$П_б = П_{валов} + V_д - V_р, \quad (27)$$

где $П_{валов}$ – прибыль до налогообложения, руб.;
$V_д$ – прочие доходы (по ценным бумагам, от долевого участия в совместных проектах и др., можно принять в размере 3% от Пр), руб.;
$V_р$ – прочие расходы (выплаты по экономическим санкциям и др., можно принять в размере 0,5% от Пр), руб.

На основании формулы (27) можно получить следующее значение:

$$П_б = П_{валов} + V_д - V_р = 110235.57 + 1805.76 - 300.96 =$$

$$= 111740.37 \text{ руб.}$$

Чистая прибыль рассчитывается по следующей формуле:

$$П_ч = П_б - Н_{пр}, \quad (28)$$

где $П_б$ – балансовая прибыль, которую может получить фирма (организация), разрабатывающая и реализующая ПО, руб.;
$Н_{пр}$ – налог на прибыль (20%), руб.

На основании формулы (28) можно получить следующее значение:

$$П_ч = П_б - Н_{пр} = 111740.37 - 22348.08 = 89392.29 \text{ руб.}$$

“Чистая” прибыль расходуется фирмой по следующим направлениям:

- на техническое развития (фонд накопления);
- на создание фонда потребления;
- на выплату дивидендов, если фирма является акционерным обществом;
- на благотворительные и экологические цели;
- на прочие цели.

Больше всего (50%) планируется расходовать чистую прибыль на фонд накопления в связи с необходимостью поддержания продукта; далее планируется отвести 30% чистой прибыли на социальные нужды, что обусловлено необходимостью проведения оздоровительных мероприятий для сотрудников фирмы, выплаты премий; на прочие цели запланировано использовать оставшиеся средства (20%).

Так как фирма не является акционерным обществом, а представляет собой группу, сформированную в рамках инженерного центра ВУЗа, то выплаты под дивидендам отсутствуют.

Так как воздействие на экологию от деятельности фирмы незначительно, выплаты на экологические цели не планируются.

Фирма не занимается благотворительной деятельностью, поэтому выплаты в этом направлении не производятся.

Таким образом, можно свести все показатели в одну таблицу (таблица 9).

Таблица 9 – Итоговый расчет формирования и использования выручки и прибыли

Наименование показателя	Обозначение	Сумма, тыс. руб.
Валовая выручка от реализации ПО (всех копий) по рыночной цене (без торговой наценки)	$ВР_{бр}$	793.694
Налог на добавленную стоимость	НДС	132.282
Выручка от продажи ПО по цене создания	$ВР_{н}$	661.412
Общие затраты на создание и копирование всех реализуемых ПО	$З_{об}$	551.176
Прибыль от продаж всех копий ПО	$П_{валов}$	110.236
Доходы от внереализационных операций	$V_{д}$	1.806
Расходы от внереализационных операций	$V_{р}$	0.301
Балансовая прибыль	$П_{б}$	111.740
Налог на прибыль	$Н_{пр}$	22.348
“Чистая” прибыль – всего	$П_{ч}$	89.392
“Чистая” прибыль – на техническое развитие		44.696
“Чистая” прибыль – на образование фонда потребления		26.818
“Чистая” прибыль – на выплату дивидендов		0
“Чистая” прибыль – на благотворительные и экологические цели		0
“Чистая” прибыль – на другие цели		17.878

4.3 Расчёт затрат, связанных с покупкой, внедрением и использованием программного обеспечения

При анализе эффективности приобретаемой компьютерной программы следует рассчитать следующие показатели:

- капитальные затраты на приобретение и внедрение ПО;
- текущие затраты пользователя, связанные с применением программы,

- экономию от использования компьютерной программы;
- срок окупаемости капитальных затрат.

Общее время эксплуатации компьютера в течение года рассчитываются по следующей формуле:

$$Т_{об} = Д_{с} * S * Д_{р} * Н_{м} * К_{ис}, \quad (29)$$

где $Д_{с}$ – длительность смены, час.;
S – число смен работы компьютера;
$Д_{р}$ – среднее число рабочих дней в месяце, дней;
$Н_{м}$ – число месяцев в году;
$К_{ис}$ – средний коэффициент использования компьютера в течение года (0.8).

На основании формулы (29) можно получить следующее значение:

$$Т_{об} = Д_{с} * S * Д_{р} * Н_{м} * К_{ис} = 8 * 1 * 21 * 12 * 0.8 = 1612.8 \text{ час.}$$

Планируется, что преподаватель будет использовать программу в сумме около 15 часов каждый месяц.

Величина затрачиваемого компьютерного времени на решение задач с помощью купленной программы рассчитывается по следующей формуле:

$$Т_{м} = В_{з}^{мес} * М, \quad (30)$$

где $В_{з}^{мес}$ – время решения задач с помощью приобретенного ПО в течение одного дня, час.;
М – количество месяцев использования купленного ПО в течение года, мес.

На основании формулы (30) можно получить следующее значение:

$$Т_{м} = В_{з}^{мес} * М = 15 * 12 = 180 \text{ час.}$$

Можно принять, что цена компьютера офисного назначения составляет 30000 рублей.

Капитальные вложения на создание рабочего места пользователя ПО (без учета износа) рассчитываются по следующей формуле:

$$К_{рм} = \frac{(S * Ц_{пл} + З_{меб}) * Т_{м}}{Т_{об}}, \quad (31)$$

где S – размер площади, которую занимают компьютерный стол и специалист, работающий с помощью компьютера, кв. м.;
$Ц_{пл}$ – рыночная цена 1 кв. м. площади на момент покупки ПО, руб.;
$З_{меб}$ – затраты на приобретение мебели (компьютерный стол, кресло

для пользователя, стол для принтера и др., их можно принять 15% от $Ц_{ком}$), руб.;

$Т_м$ – время использования компьютера в течение года для решения всех задач с помощью приобретенной программы, час.;

$Т_{об}$ – общее время эксплуатации компьютера в течение года, час.

На основании формулы (31) можно получить следующее значение:

$$К_{рм} = \frac{(S * Ц_{пл} + З_{меб}) * Т_м}{Т_{об}} = \frac{(4 * 43709 + 4500) * 180}{1612.8} =$$

$$= 20015.18 \text{ руб.}$$

Капитальные затраты на техническое оснащение рабочего места пользователя ПО рассчитываются по следующей формуле:

$$К_{тех} = \frac{(Ц_{ком} + Ц_{тех}) * (1 + К_{тр}) * (1 - К_{из}) * Т_м}{Т_{об}}, \quad (32)$$

где $Ц_{ком}$ – рыночная цена компьютера определенной модели на момент покупки программы, руб.;

$Ц_{тех}$ – цена дополнительного технического оснащения компьютера (стоимость принтера, звуковой карты, проигрывателя и др., их можно принять в размере 50% от $Ц_{ком}$), руб.;

$К_{тр}$ – коэффициент, учитывающий затраты на транспортировку и отладку компьютера и других технических средств; рекомендуется принять в размере 0,01-0,05 от $Ц_{ком}$ (данные расходы определяются в том случае, если приобретается новая ПЭВМ. Если же будет использоваться действующий компьютер, то $К_{тр}$ = 0);

$К_{из}$ – коэффициент, учитывающий степень износа действующего компьютера, на котором будут решаться задачи с помощью купленного ПО; его можно определить укрупненно, путем сопоставления фактического и проектного (обычно не более 5 лет) сроков службы ПЭВМ; в случае приобретения нового компьютера $К_{из}$ = 0.;

$Т_м$ – время использования компьютера в течение года для решения всех задач с помощью приобретенной программы, час.;

$Т_{об}$ – общее время эксплуатации компьютера в течение года, час.

На основании формулы (32) можно получить следующее значение:

$$К_{тех} = \frac{(Ц_{ком} + Ц_{тех}) * (1 + К_{тр}) * (1 - К_{из}) * Т_м}{Т_{об}} =$$

$$= \frac{(30000 + 15000) * (1 + 0) * \left(1 - \frac{3}{5}\right) * 180}{1612.8} = 2008.93 \text{ руб.}$$

Капитальные затраты на приобретение и внедрение ПО рассчитываются по следующей формуле:

$$К_{по} = Ц_{по} + К_{рм} + К_{тех} + К_{пр}, \quad (33)$$

где $Ц_{по}$ – затраты на покупку ПО (принимаются равными рыночной цене программы), руб.;

$К_{рм}$ – капитальные вложения на создание рабочего места пользователя ПО, руб.;

$К_{тех}$ – капитальные вложения на техническое оснащение рабочего места пользователя ПО, руб.;

$К_{пр}$ – прочие капитальные вложения, связанные с внедрением ПО (10% от $Ц_{по}$), руб.

На основании формулы (33) можно получить следующее значение:

$$К_{по} = Ц_{по} + К_{рм} + К_{тех} + К_{пр} = 8730.64 + 20015.18 + 2008.93 +$$

$$+873.06 = 31627.81 \text{ руб.}$$

Таким образом, можно представить данные в виде таблицы (таблица 10).

Таблица 10 – Капитальные затраты на покупку и внедрение ПО

Наименование затрат	Обозначение	Сумма, тыс. руб.
Затраты на покупку ПО	$Ц_{по}$	8.731
Затраты на создание рабочего места	$К_{рм}$	20.015
Затраты на техническое оснащение рабочего места	$К_{тех}$	2.009
Прочие капитальные затраты	$К_{пр}$	0.873
Итого	$К_{по}$	31.628

Годовые текущие затраты пользователя, связанные с применением программы, рассчитываются по следующей формуле:

$$З_{тек} = Т_{м} * С_{м} + \frac{Ц_{по}}{Т_{с}}, \quad (34)$$

где $Т_{м}$ – время использования компьютера в течение года для решения всех задач с помощью приобретенной программы, час.;

$С_м$ – стоимость одного часа эксплуатации компьютера, руб.;

$Ц_{по}$ – затраты на покупку ПО (принимаются равными рыночной цене программы), руб.;

$Т_с$ – планируемый срок использования приобретенной компьютерной программы (с учетом морального износа не более 5 лет), час.

На основании формулы (34) можно получить следующее значение:

$$З_{тек} = Т_м * С_м + \frac{Ц_{по}}{Т_с} = 180 * 111.55 + \frac{8730.64}{3} =$$

$$= 22989.21 \text{ руб.}$$

Стоимость одного часа эксплуатации компьютера взята как стоимость одного часа эксплуатации компьютера разработчика, так как в обоих случаях считается, что будет задействован компьютер офисного назначения со сходными характеристиками.

При решении задачи без использования разработанной программы потребуется участие двух человек: преподавателя, составляющего требования к ВЛС и математическую модель ВЛС, и программиста, разрабатывающего ВЛС на основании требований и математической модели.

В этом случае можно считать, что общее время работы над созданием ВЛС увеличится минимум в 2 раза в связи с увеличением объема работ, причем 50-80% времени уйдет на разработку ВЛС.

Время, затраченное преподавателем на разработку ВЛС без использования разработанной программы, равно:

$$В_{преп}^{влс} = 180 * 2 * 0.5 = 180 \text{ час.}$$

Время, затраченное программистом на разработку ВЛС без использования разработанной программы, равно:

$$В_{п}^{влс} = 180 * 2 * 0.5 = 180 \text{ час.}$$

Фонд оплаты труда работников, участвующих в решении задач без использования разработанной программы, рассчитанный по минимальному уровню зарплаты, можно найти, основываясь на формулах (6)-(9):

$$ФОТ_м = \frac{Р_{преп} * В_{преп}^{влс} * О_м^{преп} + Р_п * В_п^{влс} * О_м^п}{Д_с * Д_р} + П + В_{рк} =$$

$$= \frac{1 * 180 * 12130 + 1 * 180 * 12130}{8 * 21} + 5198.57 + 4678.71 =$$

$$= 35870.14 \text{ руб.}$$

Затраты на решение задач без применения компьютерной программы рассчитываются по следующей формуле:

$$З_р = \frac{Р * В_{уч} * О_м}{Д_с * Д_р} + П + В_{рк} + Е_{сн}, \quad (35)$$

где Р – число работников, участвующих в решении задач ручным способом, чел.;

$В_{уч}$ – время участия каждого работника в решении задач ручным способом в течение года, час.;

$О_м$ – месячный оклад работника в соответствии с его категорией или тарифным разрядом, руб.;

П – премия, предусмотренная для работников, участвующих в решении задач ручным способом (20% от $ЗП_{об}$), руб.;

$В_{рк}$ – выплаты по районному коэффициенту (установлены для г. Кирова в размере 15% от ($ЗП_{об}$ + П)), руб.;

$Е_{сн}$ – единый социальный налог (30%);

$Д_с$ – длительность смены, час.;

$Д_р$ – среднее число рабочих дней в месяце, дней.

На основании формулы (35) можно получить следующее значение:

$$З_р = \frac{Р * В_{уч} * О_м}{Д_с * Д_р} + П + В_{рк} + Е_{сн} =$$

$$= \frac{2 * 180 * 12130}{8 * 21} + 5198.57 + 4678.71 + 7797.86 = 43668 \text{ руб.}$$

Годовая экономия на текущих расходах, которую может получить фирма от применения программного обеспечения, рассчитывается по следующей формуле:

$$Э_г = З_р - З_{тек}, \quad (36)$$

где $З_р$ – затраты на решение задач, действующим способом (обычно ручным), руб.;

$З_{тек}$ – годовые текущие затраты пользователя, связанные с применением программы, руб.

На основании формулы (36) можно получить следующее значение:

$$Э_г = З_р - З_{тек} = 43668 - 22989.21 = 20678.79 \text{ руб.}$$

Срок окупаемости капитальных затрат на покупку и внедрение компьютерной программы рассчитывается по следующей формуле:

$$Т_{ок} = \frac{К_{по}}{Э_г}, \quad (37)$$

где $К_{по}$ – капитальные затраты на приобретение и внедрение ПО, руб.;

$Э_г$ – годовая экономия на текущих расходах, которую может получить фирма от применения программного обеспечения, руб.

На основании формулы (37) можно получить следующее значение:

$$Т_{ок} = \frac{К_{по}}{Э_г} = \frac{31627.81}{20678.79} = 1.5 \text{ лет.}$$

4.4 Выводы к разделу 4

Итоговые значения расчетов приведены в таблице ниже (таблица 11).

Таблица 11 – Финансово-экономические показатели создания и использования ПО

Наименование показателя	Единица измерения	Значение показателя
1 Показатели фирмы-разработчика ПО		
1.1 Число специалистов, участвующих в разработке компьютерной программы	чел.	2
1.2 Время создания ПО	чел.мес.	4.89
1.3 Число копий ПО, предполагаемых к реализации	шт.	100
1.4 Затраты на создание ПО	тыс.руб.	300.960
1.5 Розничная цена одной копии	тыс.руб.	8.731
1.6 Уровень рентабельности	%	20
1.7 Балансовая прибыль от продажи ПО	тыс.руб.	111.740
1.8 "Чистая" прибыль	тыс.руб.	89.392
2. Показатели фирмы-покупателя ПО		
2.1 Капитальные затраты на покупку и внедрение ПО	тыс.руб.	31.628
2.2 Годовые текущие расходы, связанные с использованием ПО	тыс.руб.	22.989
2.3 Годовая экономия от применения ПО	тыс.руб.	20.679
2.4 Расчетный срок окупаемости капитальных затрат	лет	1.5

На основании данных таблицы 11 можно сделать вывод, что создание и приобретение программы экономически выгодно, капитальные затраты на покупку и внедрение продукта окупятся за 1.5 года.

Заключение

В ходе данной выпускной квалификационной работы был разработан модуль проектирования виртуальных лабораторных стендов с использованием языка программирования C#. Весь процесс разработки был разбит на несколько последовательных этапов для достижения конечной цели: обзора предметной области виртуальных лабораторных стендов, проектирования модуля, программной реализации и технико-экономического обоснования разработки модуля.

Обзор предметной области и сравнение аналогов позволили выделить конкурентные особенности, которые были далее реализованы в модуле: разграничение логики и интерфейса, наличие визуального языка программирования, удобного в использовании и обладающего достаточным функционалом для проектирования моделей.

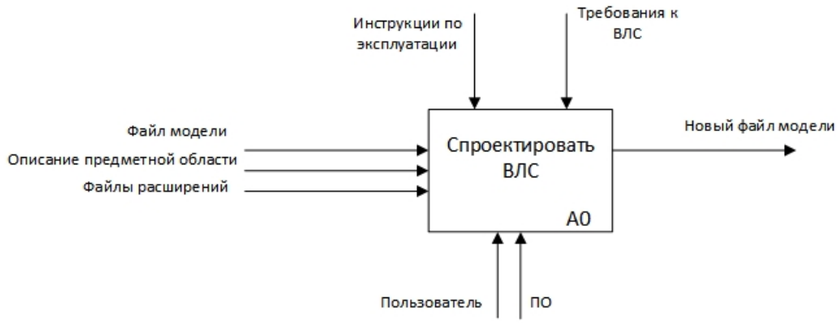
В ходе проектирования была определена концепция стенда, разработаны функциональные модели, модели данных, произведено проектирование с использованием языка UML. Особое внимание было уделено разработке нижнего уровня визуального языка программирования LabScript.

Программная реализация модуля была продемонстрирована экранными формами графического пользовательского интерфейса и фрагментами исходного кода модуля. Также в дополнение к программной реализации был разработан демонстрационный стенд «Модель температуры микроклимата теплицы» с целью отражения практической применимости разработанного модуля.

Было произведено экономическое обоснование разработки программного продукта: расчет затрат на создание продукта, розничной цены одной копии продукта, выручки и прибыли от реализации продукции, расходов от внедрения и эксплуатации программы. В результате было выяснено, что затраты пользователя на покупку и внедрение решения окупятся за 1.5 года. Причем цена модуля оказалась ниже цен за аналоги, что является также достоинством модуля.

Следует отметить, что разработка данного модуля является лишь первым шагом к созданию автоматизированной системы проектирования, которая позволит преподавателям повысить качество подачи учебного материала за счет использования разработанных в соответствии с требованиями учебного курса и с небольшими затратами времени и средств виртуальными лабораторными стендами.

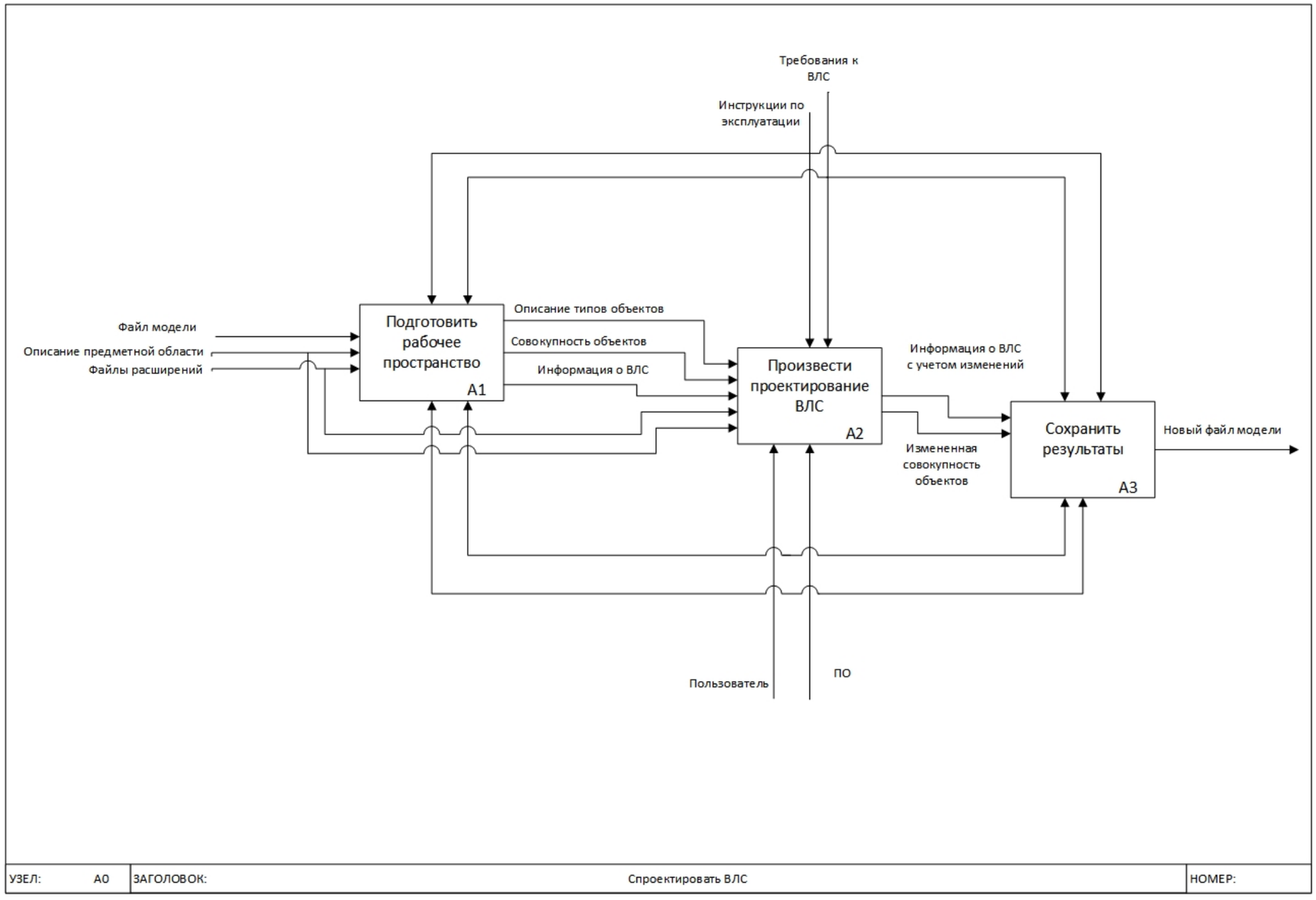
Приложение А
(обязательное)
Модель модуля в нотации IDEF0



УЗЕЛ: A0	ЗАГОЛОВОК: Спроектировать ВЛС	НОМЕР:

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Контекстная диаграмма IDEF0	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 82	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

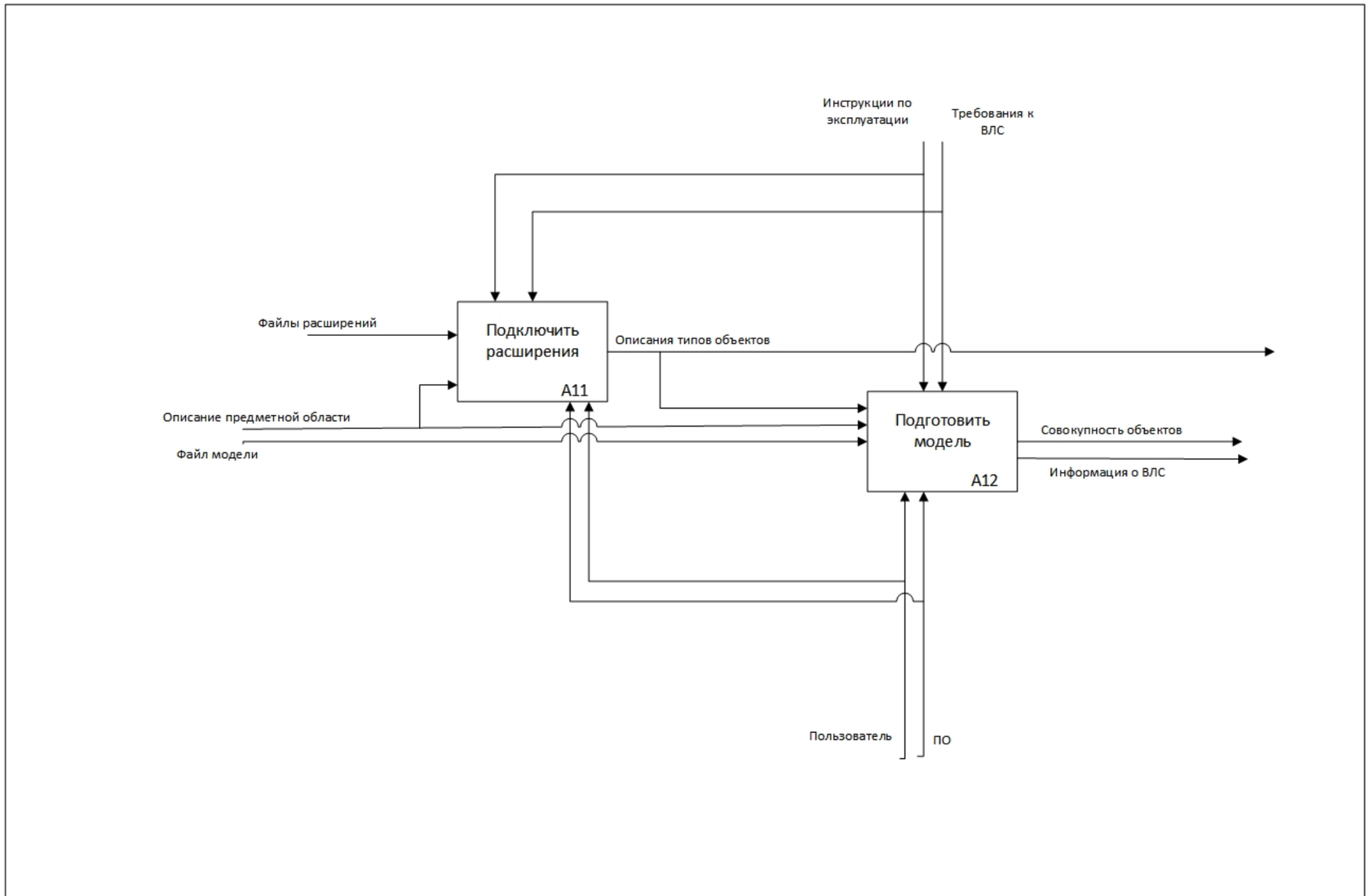


					ТПЖА 090302.036 ПЗ
Изм.	Лист	№ докум.	Подпись	Дата	Диаграмма декомпозиции IDEF0
Разраб.		Дождиков И.С.			
Пров.		Ланских Ю.В.			
Т.контр.		Ланских Ю.В.			
Н.контр.		Ланских Ю.В.			
Утв.		Ланских Ю.В.			

Лит.	Масса	Масштаб
Лист 83	Листов 150	

Кафедра САУ
Группа ИТб-4301-01-00

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.



Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма «Подготовить рабочее пространство»	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 84	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение

Справ. №

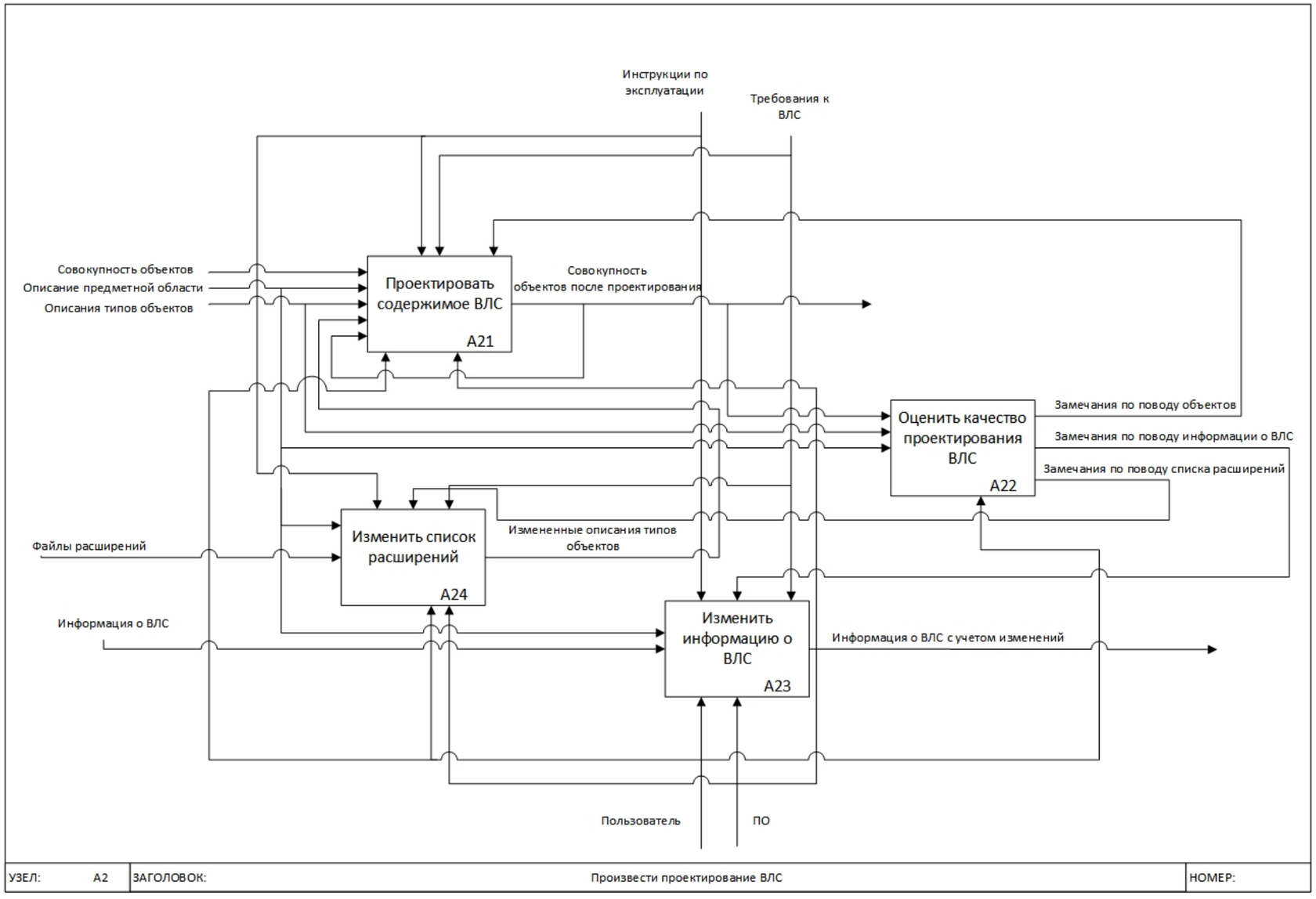
Подп. и дата

Инв. № дубл.

Взам. инв. №

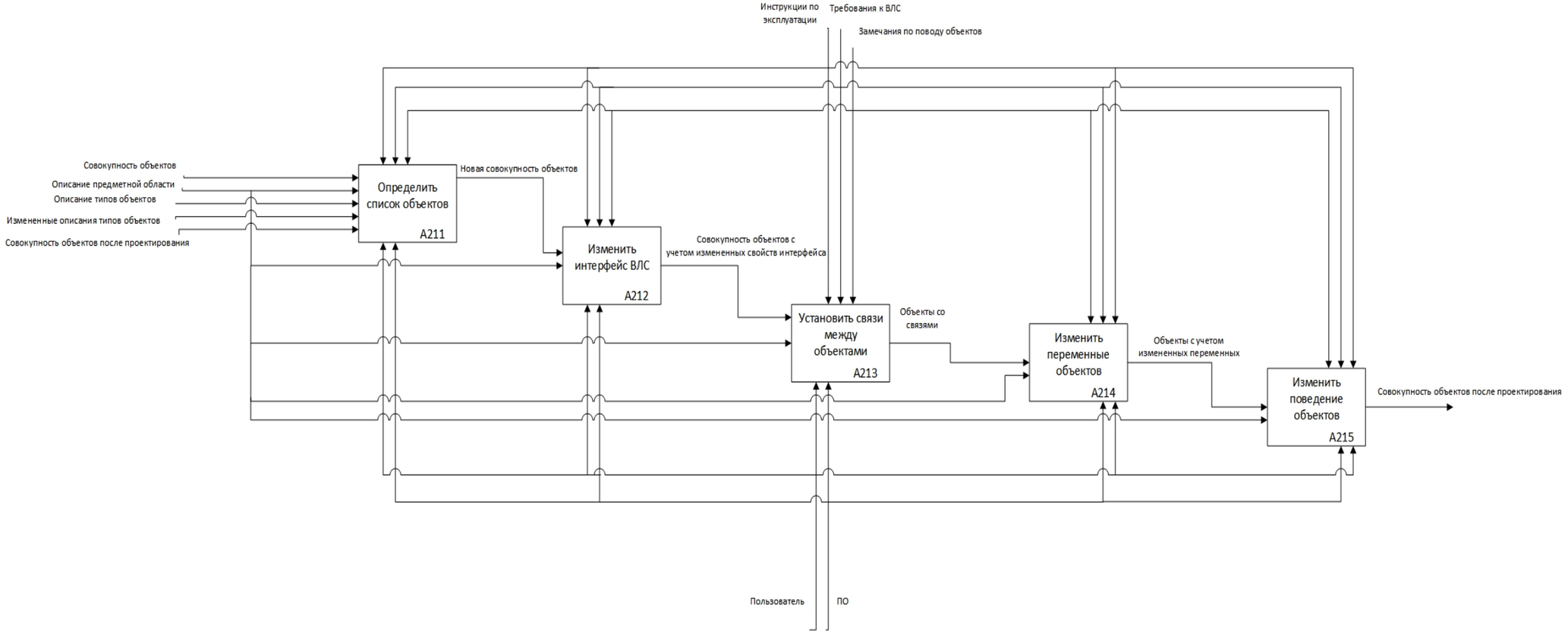
Подп. и дата

Инв. № подл.



Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Диаграмма «Произвести проектирование ВЛС»	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 85	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

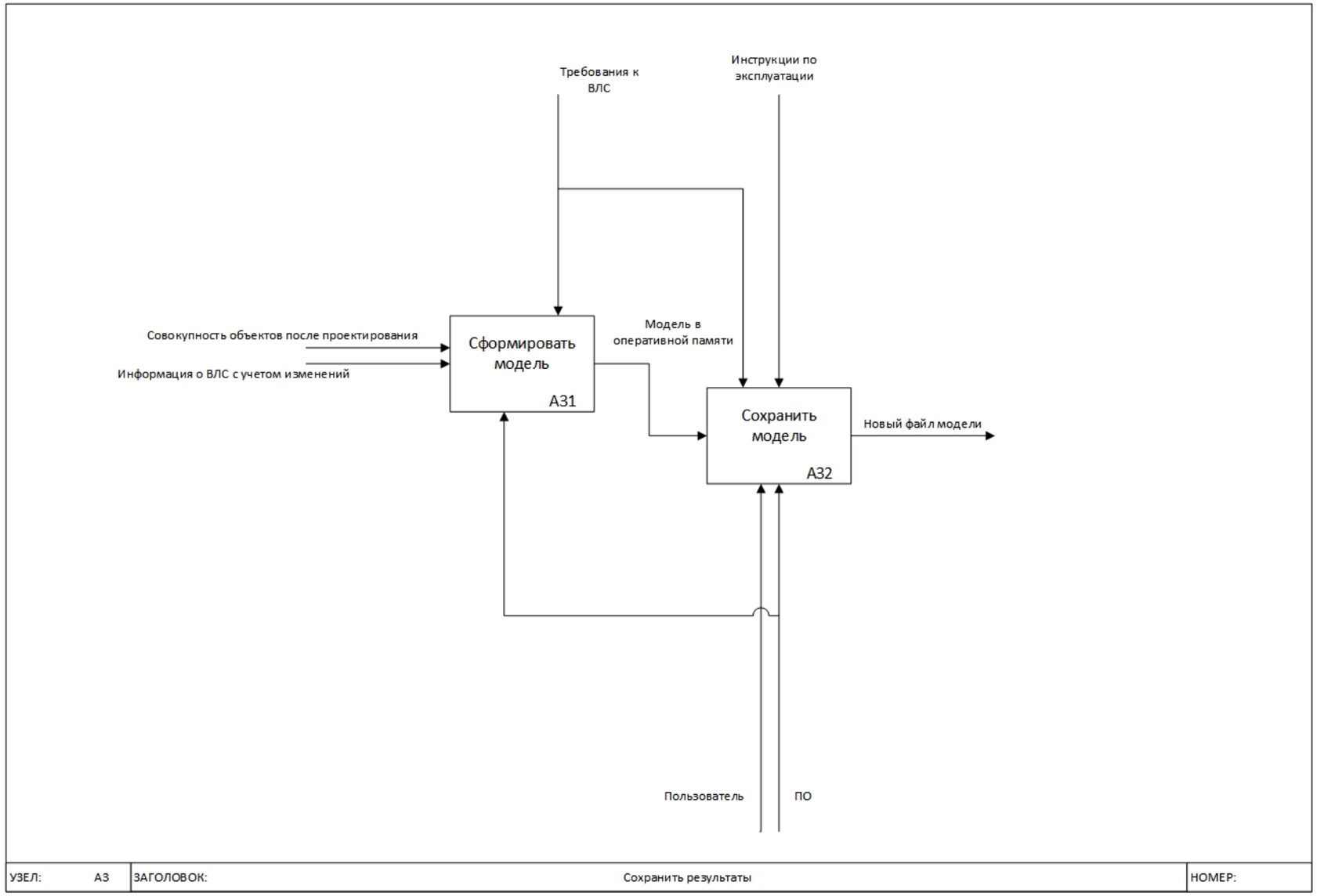
Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.



УЗЕЛ:	A21	ЗАГОЛОВОК:	Проектировать содержимое ВЛС	НОМЕР:

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Диаграмма «Проектировать содержимое ВЛС»	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 86	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение | Справ. № | Подл. и дата | Инв. № дубл. | Взам. инв. № | Подл. и дата | Инв. № подл.



Изм	Лист	№ докум.	Подпись	Дата
Разраб.		Дождиков И.С.		
Пров.		Ланских Ю.В.		
Т.контр.		Ланских Ю.В.		
Н.контр.		Ланских Ю.В.		
Утв.		Ланских Ю.В.		

ТПЖА 090302.036 ПЗ

Диаграмма «Сохранить результаты»

Лит.	Масса	Масштаб

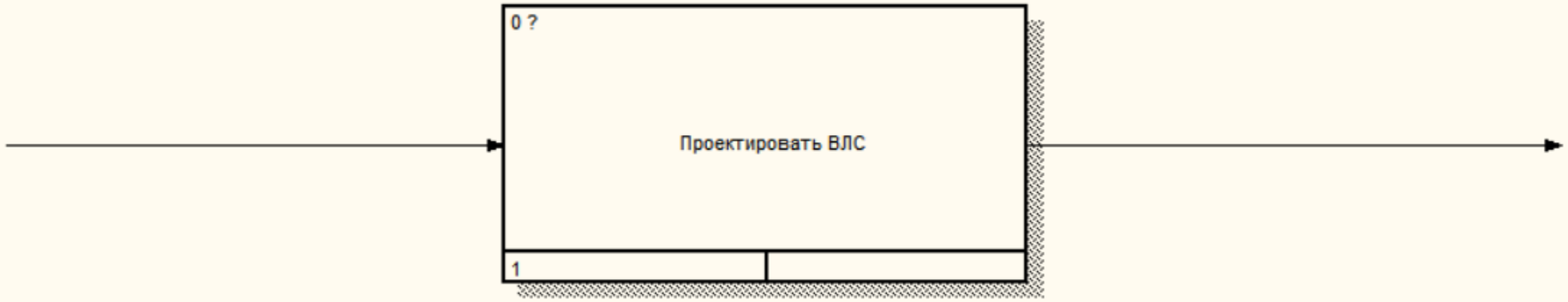
Лист 87 | Листов 150

Кафедра САУ
Группа ИТб-4301-01-00

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Приложение Б
(обязательное)
Модель модуля в нотации IDEF3

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleIDEF3	REV: 21.05.2020		DRAFT			TOP
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			



NODE:	TITLE:	NUMBER:
1	Проектировать ВЛС	

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Контекстная диаграмма IDEF3	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 89	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение

Справ. №

Подп. и дата

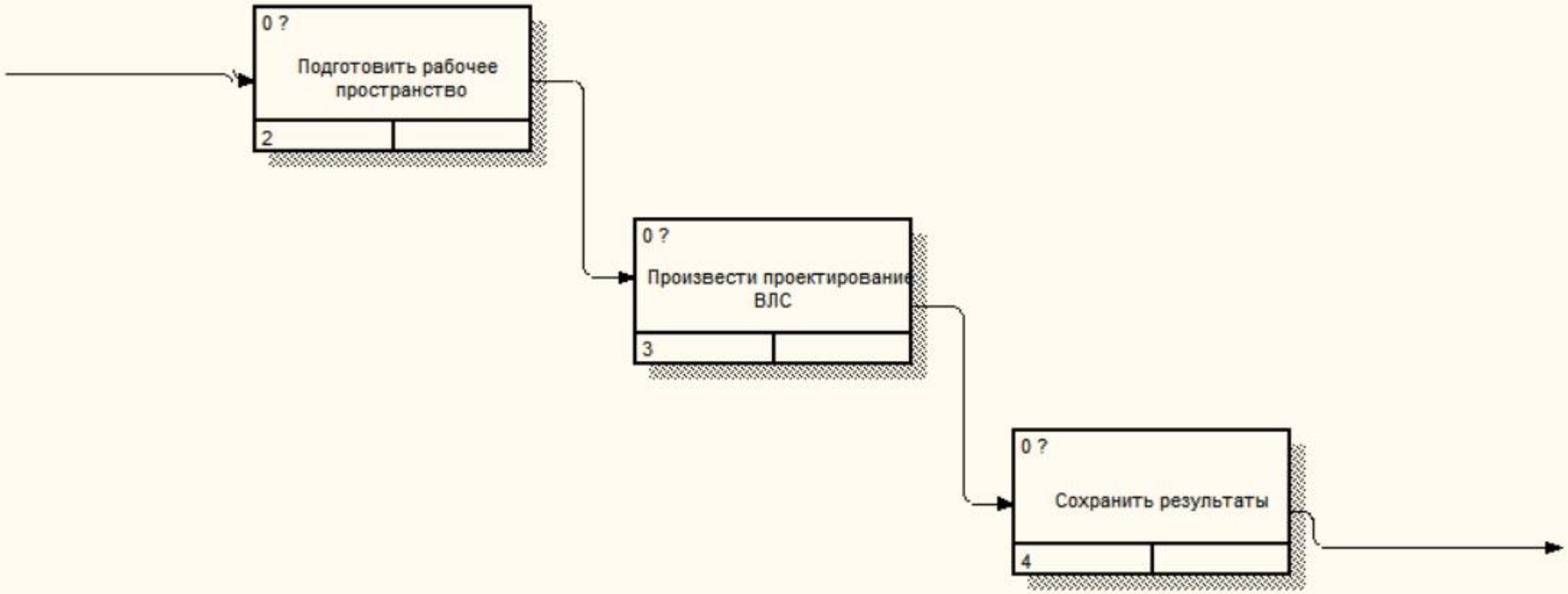
Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020	WORKING	READER DATE	CONTEXT:
	PROJECT: DesignModuleIDEF3	REV: 21.05.2020	DRAFT		
			RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		1

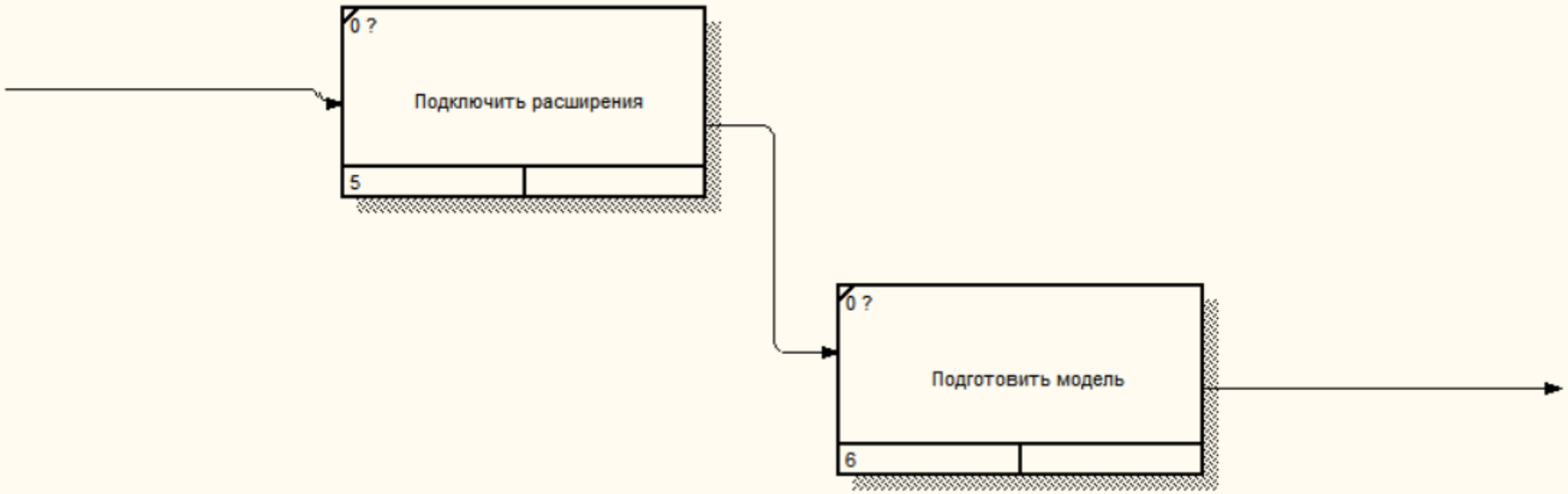


NODE:	TITLE:	NUMBER:
1.1	Проектировать ВЛС	

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма декомпозиции IDEF3	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 90	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleIDEF3	REV: 21.05.2020		DRAFT			
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			1.1



NODE:	TITLE:	NUMBER:
2.1	Подготовить рабочее пространство	

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма «Подготовить рабочее пространство»	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 91	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение

Справ. №

Подп. и дата

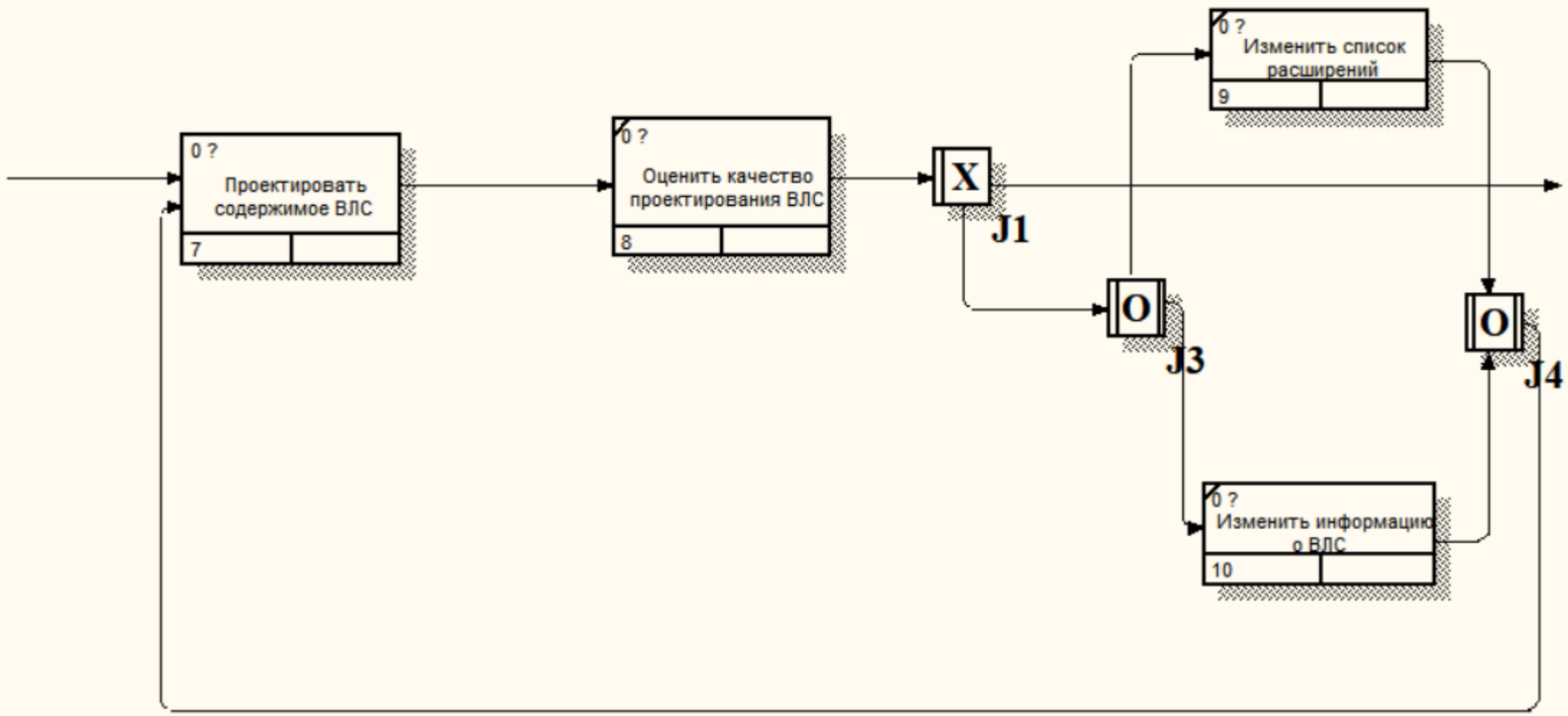
Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleIDEF3	REV: 21.05.2020		DRAFT			
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			1.1

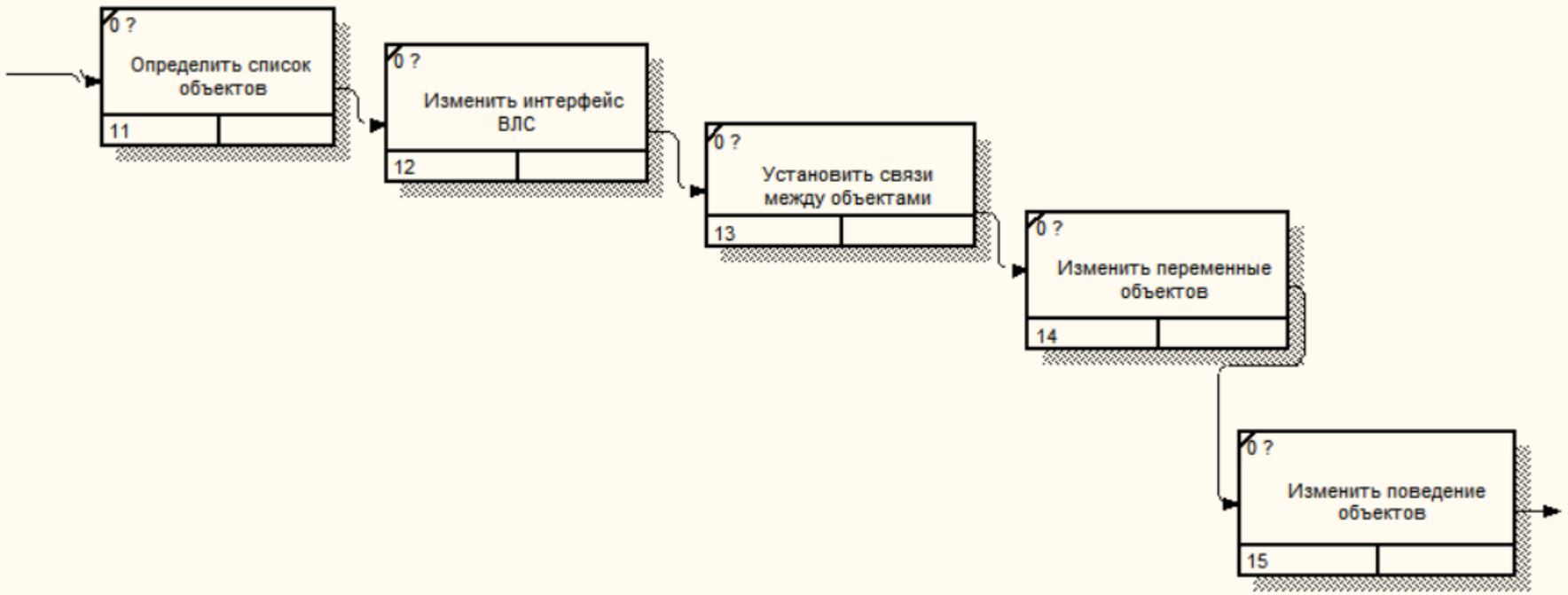


NODE:	TITLE:	NUMBER:
3.1	Произвести проектирование ВЛС	

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Диаграмма «Произвести проектирование ВЛС»	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 92	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleIDEF3	REV: 21.05.2020		DRAFT			
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			3.1

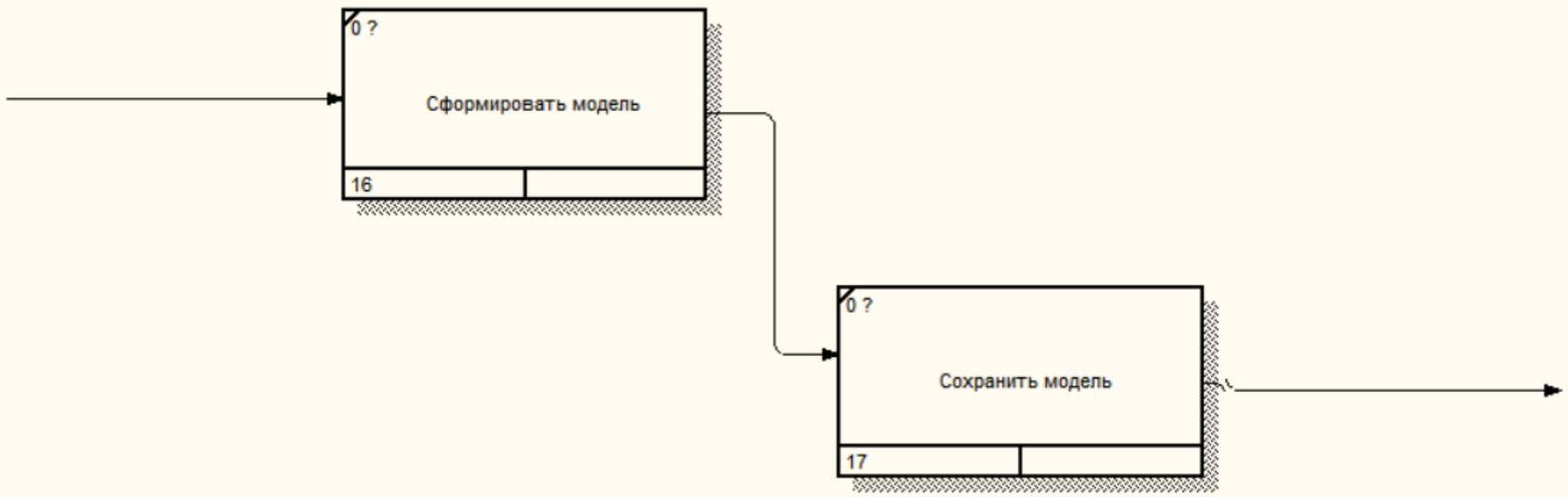


NODE:	TITLE:	NUMBER:
7.1	Проектировать содержимое ВЛС	

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Диаграмма «Проектировать содержимое ВЛС»	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 93	Листов 150
Т.контр.		Ланских Ю.В.					
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Утв.		Ланских Ю.В.					

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleIDEF3	REV: 21.05.2020		DRAFT			
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			1.1



NODE:	TITLE:	NUMBER:
4.1	Сохранить результаты	

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма «Сохранить результаты»	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 94	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение

Справ. №

Подп. и дата

Инв. № дубл.

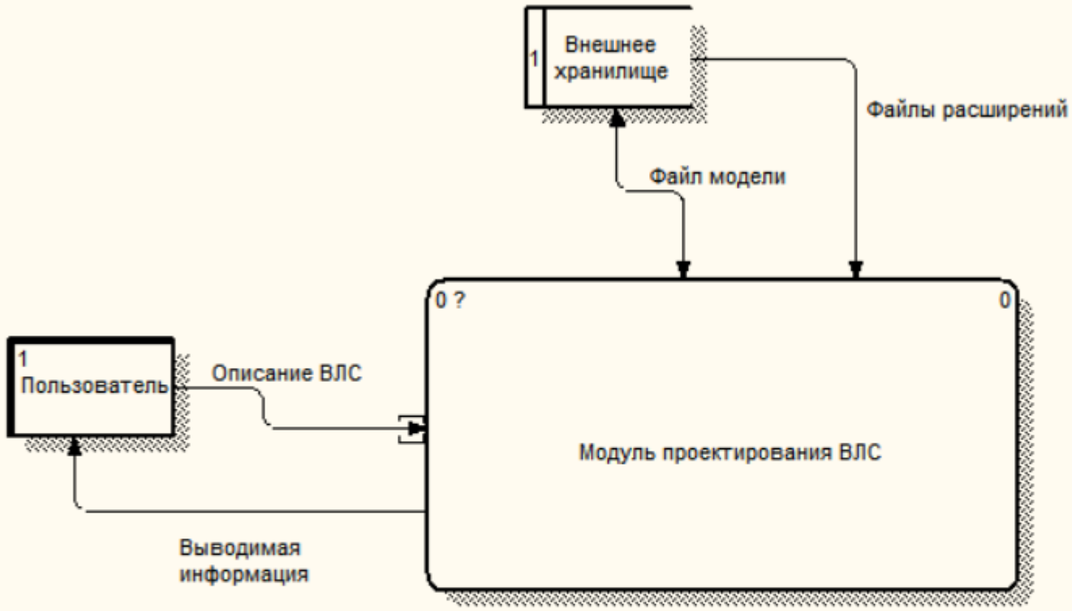
Взам. инв. №

Подп. и дата

Инв. № подл.

Приложение В
(обязательное)
Модель модуля в нотации DFD

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleDFD	REV: 21.05.2020		DRAFT			TOP
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			

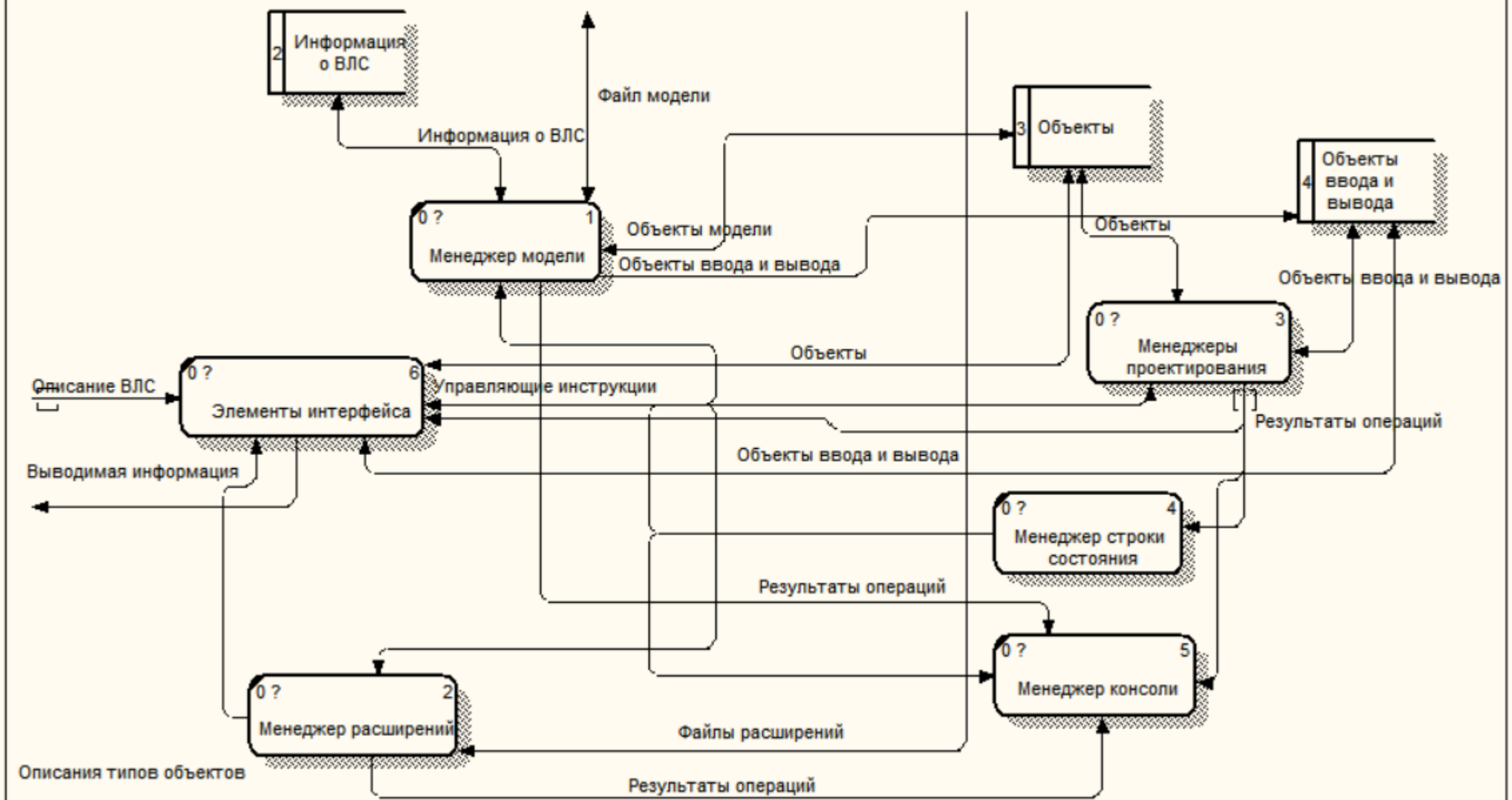


NODE:	TITLE:	NUMBER:
A-0	Модуль проектирования ВЛС	

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Контекстная диаграмма DFD	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 96	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

USED AT:	AUTHOR: Дождиков И.С.	DATE: 21.05.2020	■ WORKING	READER DATE	CONTEXT:
	PROJECT: DesignModuleDFD	REV: 21.05.2020	DRAFT		
			RECOMMENDED		■
	NOTES: 1 2 3 4 5 6 7 8 9 10		PUBLICATION		A-0

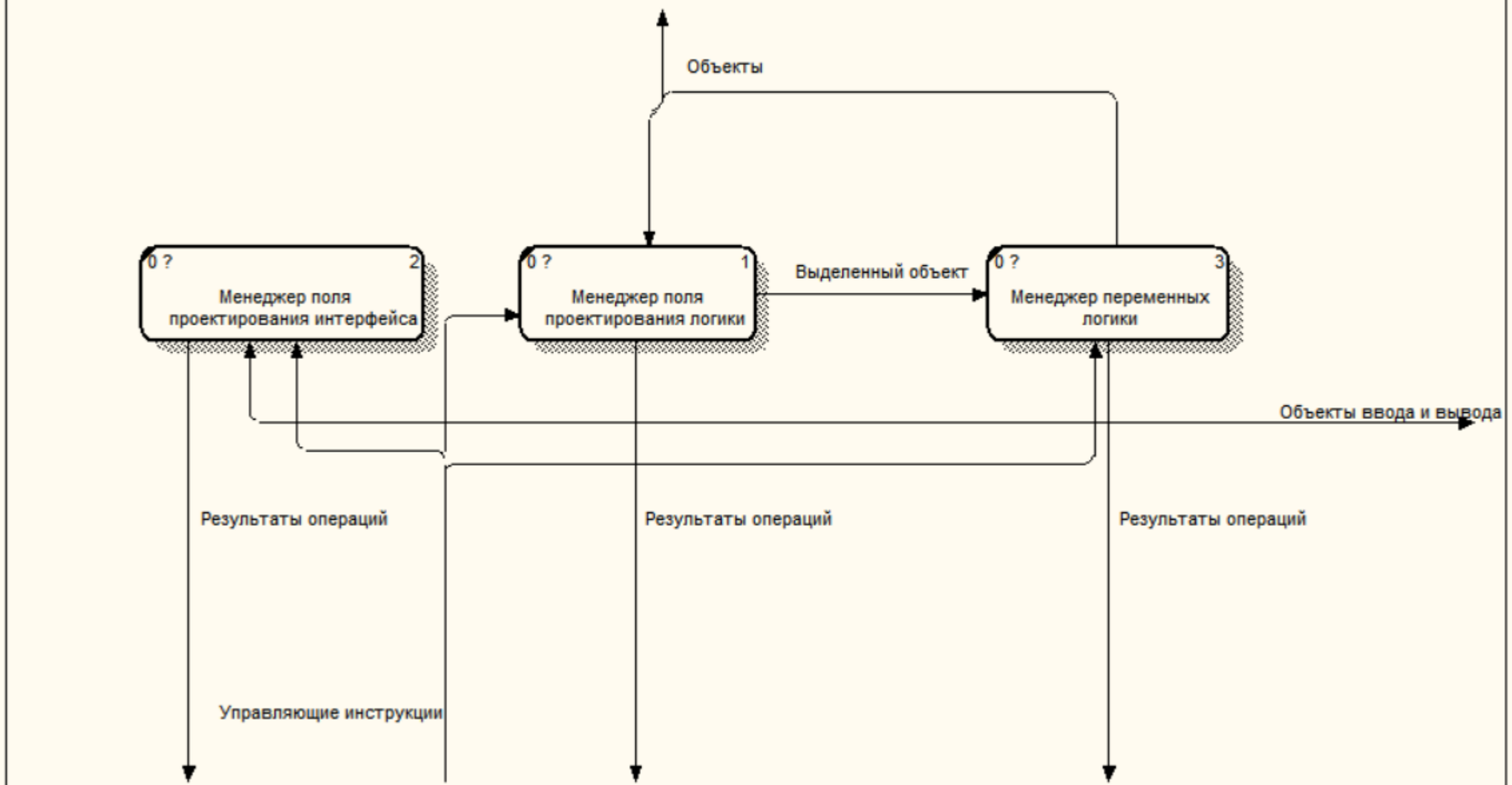


NODE:	TITLE:	NUMBER:
A0	Модуль проектирования ВЛС	

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма декомпозиции DFD	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 97	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

USED AT:	AUTHOR:	DATE: 21.05.2020		WORKING	READER	DATE	CONTEXT:
	PROJECT: DesignModuleDFD	REV: 21.05.2020		DRAFT			
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			A0



NODE:	TITLE:	NUMBER:
A3	Менеджеры проектирования	

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма «Менеджеры проектирования»	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 98	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение

Справ. №

Подп. и дата

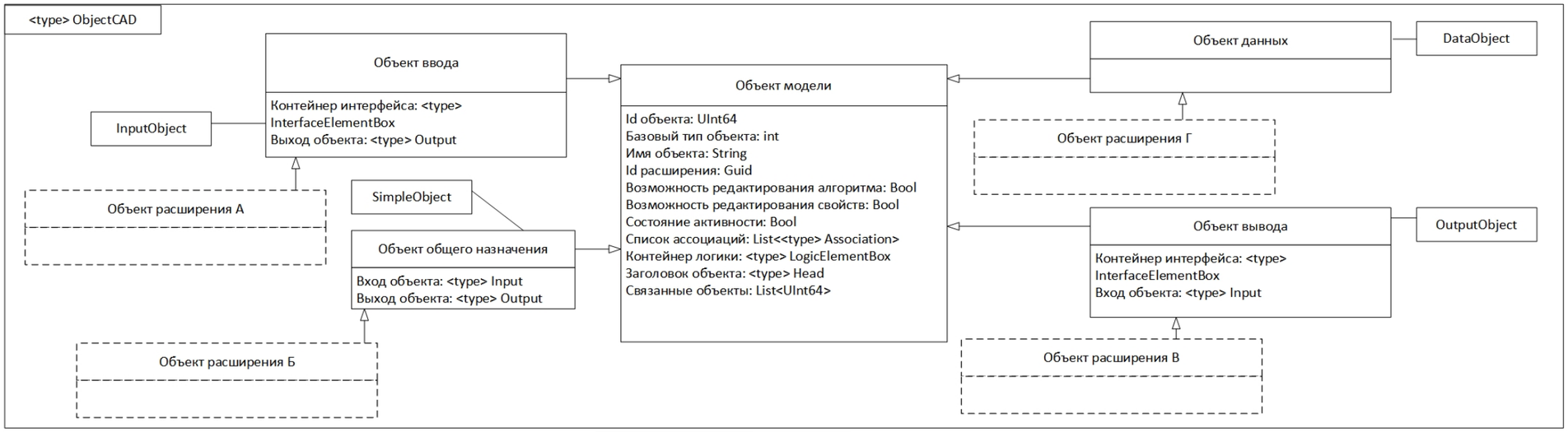
Инв. № дубл.

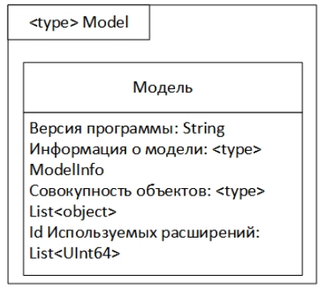
Взам. инв. №

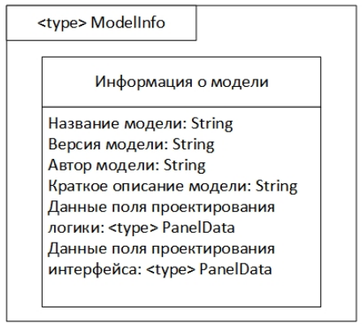
Подп. и дата

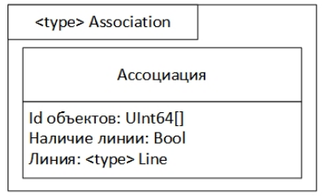
Инв. № подл.

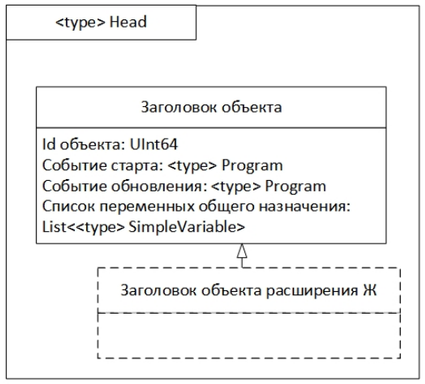
Приложение Г
(обязательное)
Диаграмма сериализации

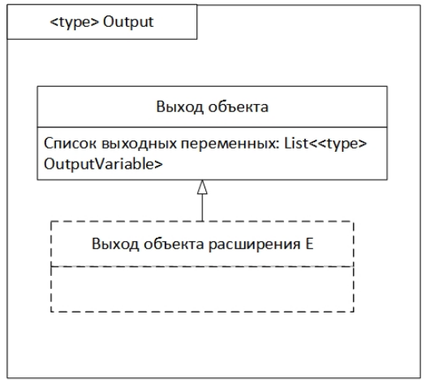


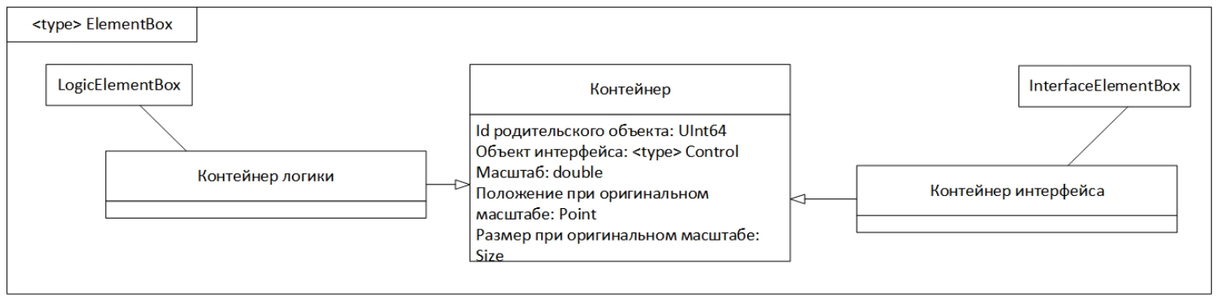


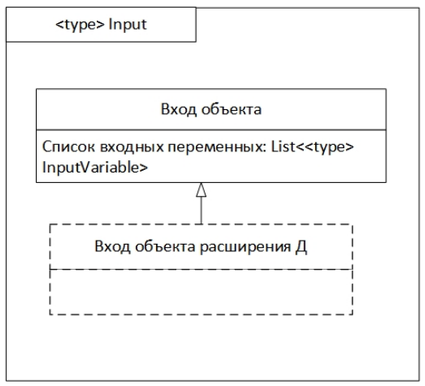


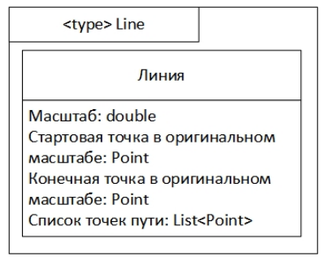












Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ			
Разраб.		Дождиков И.С.			Диаграмма сериализации (часть 1)	Лит.	Масса	Масштаб
Пров.		Ланских Ю.В.						
Т.контр.		Ланских Ю.В.				Лист 100	Листов 150	
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00		
Утв.		Ланских Ю.В.						

Перв. применение

Справ. №

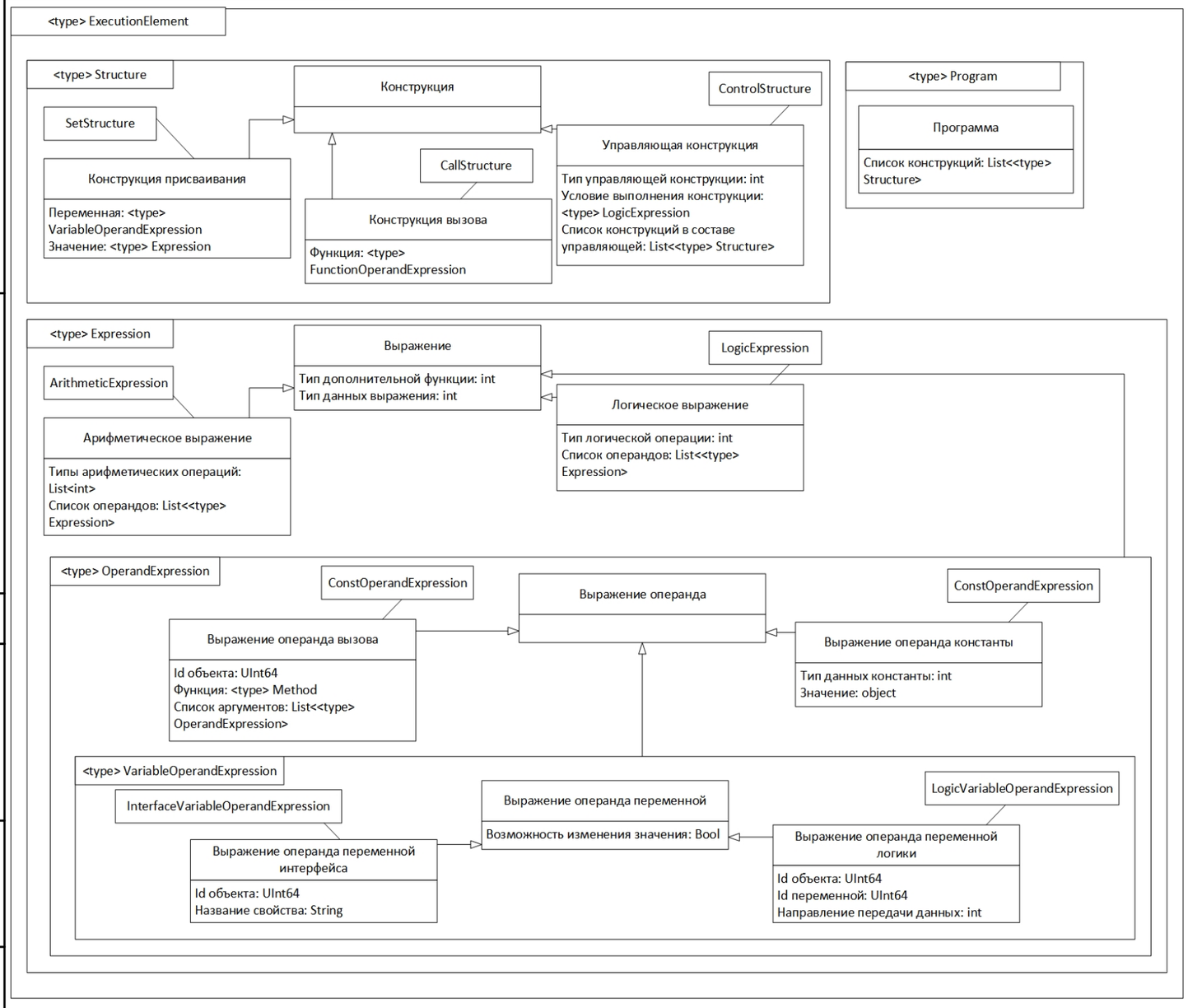
Подп. и дата

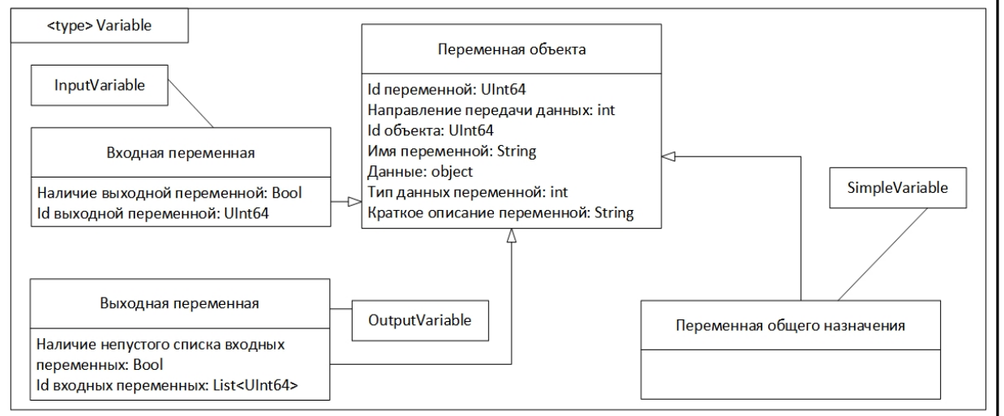
Инв. № дубл.

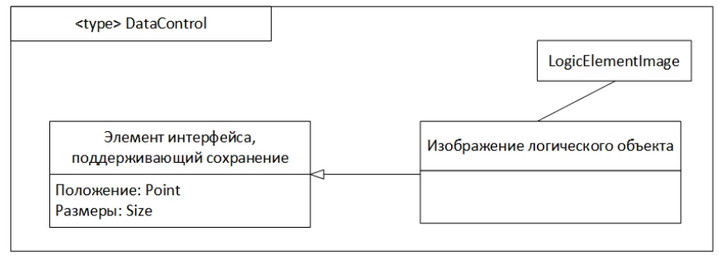
Взам. инв. №

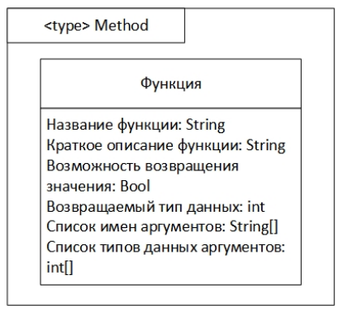
Подп. и дата

Инв. № подл.









					ТПЖА 090302.036 ПЗ
Изм.	Лист	№ докум.	Подпись	Дата	
Разраб.		Дождиков И.С.			Диаграмма сериализации (часть 2)
Пров.		Ланских Ю.В.			
Т.контр.		Ланских Ю.В.			
Н.контр.		Ланских Ю.В.			
Утв.		Ланских Ю.В.			

Лит.	Масса	Масштаб
Лист 101	Листов 150	

Кафедра САУ
Группа ИТб-4301-01-00

Перв. применение

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Приложение Д
(обязательное)
Диаграммы активности

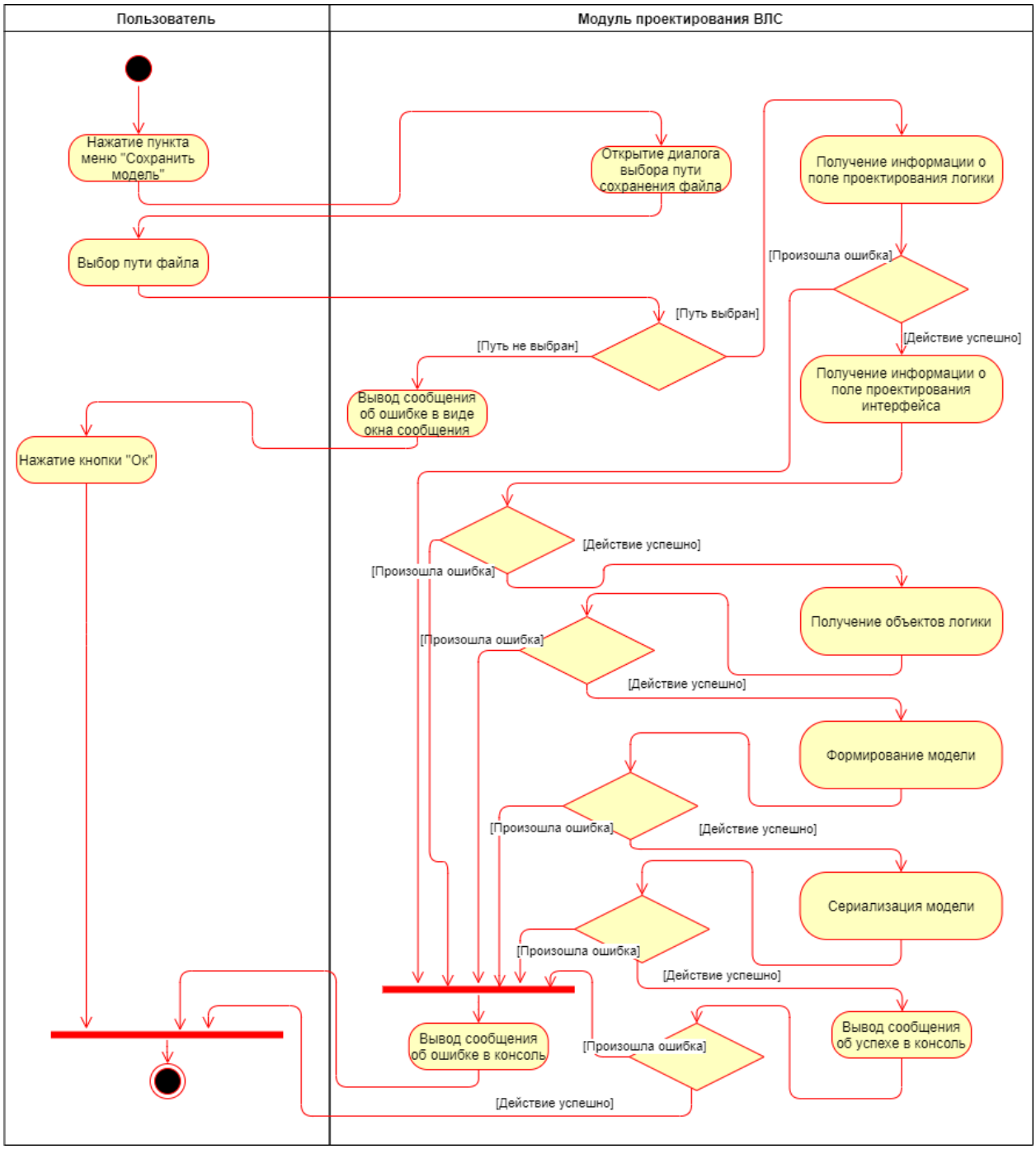


Рисунок Д.1 – Сохранить модель

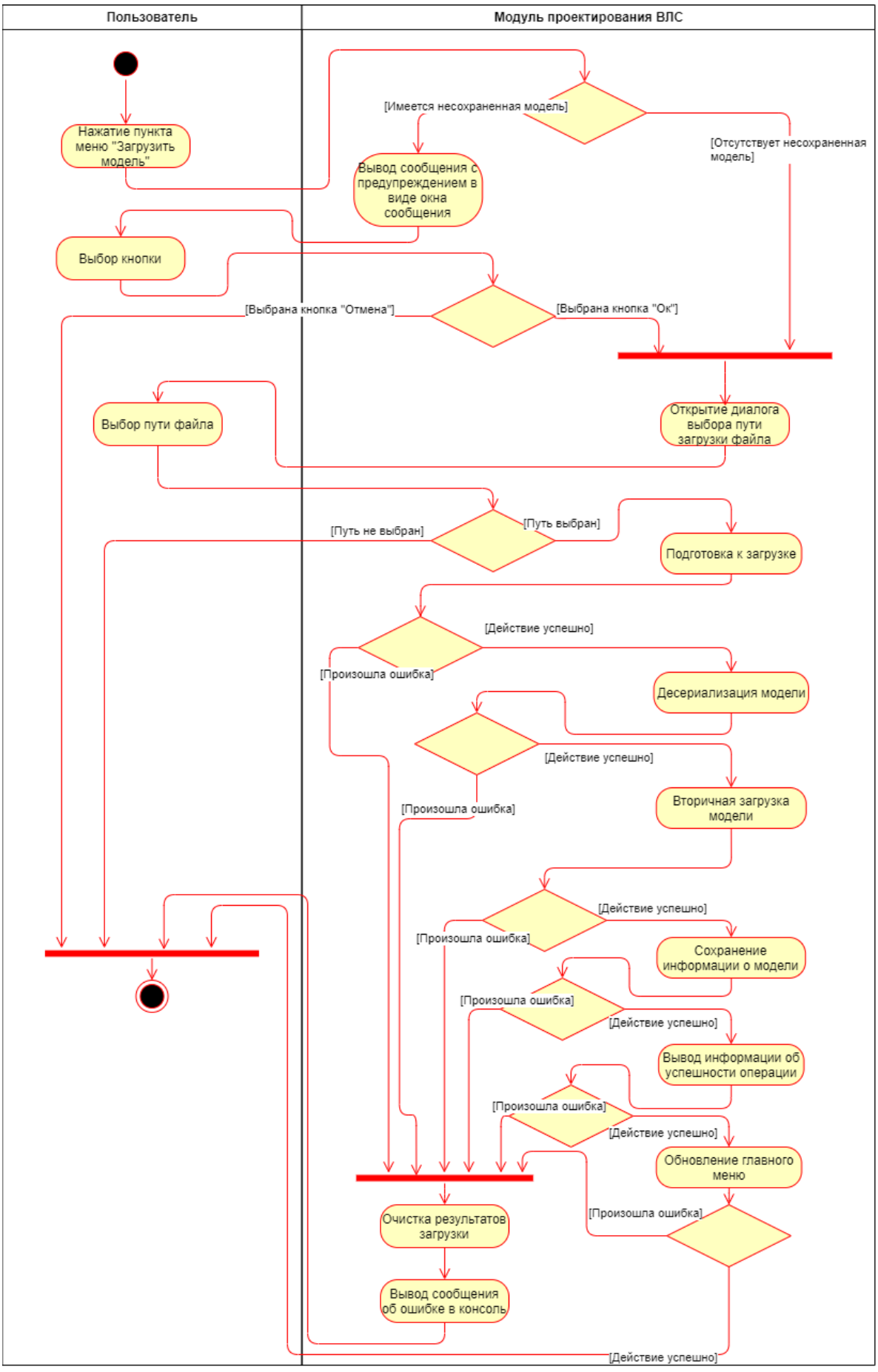


Рисунок Д.2 – Загрузить модель

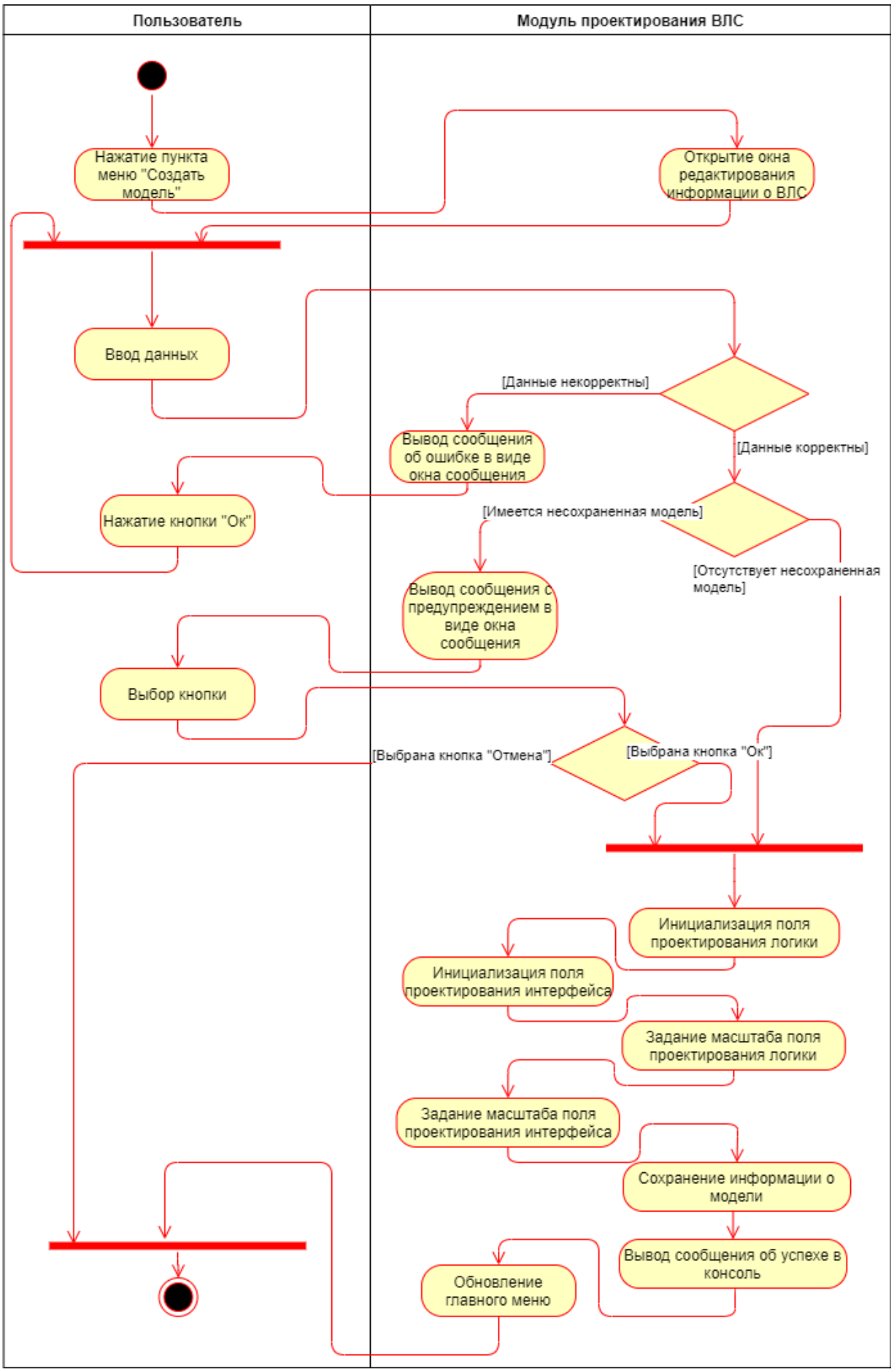
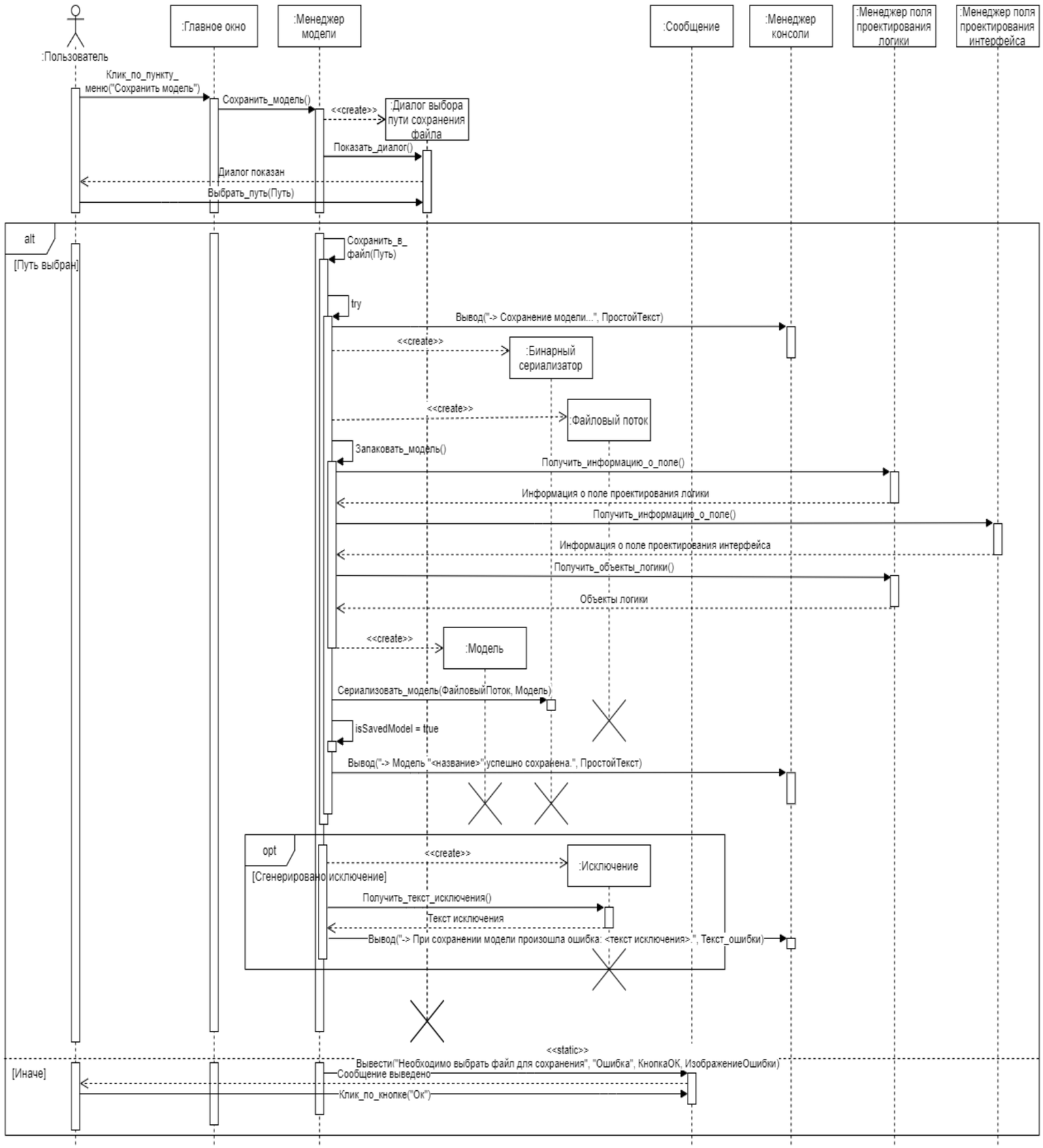


Рисунок Д.3 – Создать модель

Приложение Ж
(обязательное)
Диаграммы последовательности



					ТПЖА 090302.036 ПЗ
Изм.	Лист	№ докум.	Подпись	Дата	Диаграмма «Сохранить модель»
Разраб.		Дождиков И.С.			
Пров.		Ланских Ю.В.			
Т.контр.		Ланских Ю.В.			
Н.контр.		Ланских Ю.В.			
Утв.		Ланских Ю.В.			

Лит.	Масса	Масштаб
Лист 106	Листов 150	

Кафедра САУ
Группа ИТб-4301-01-00

Перв. применение

Справ. №

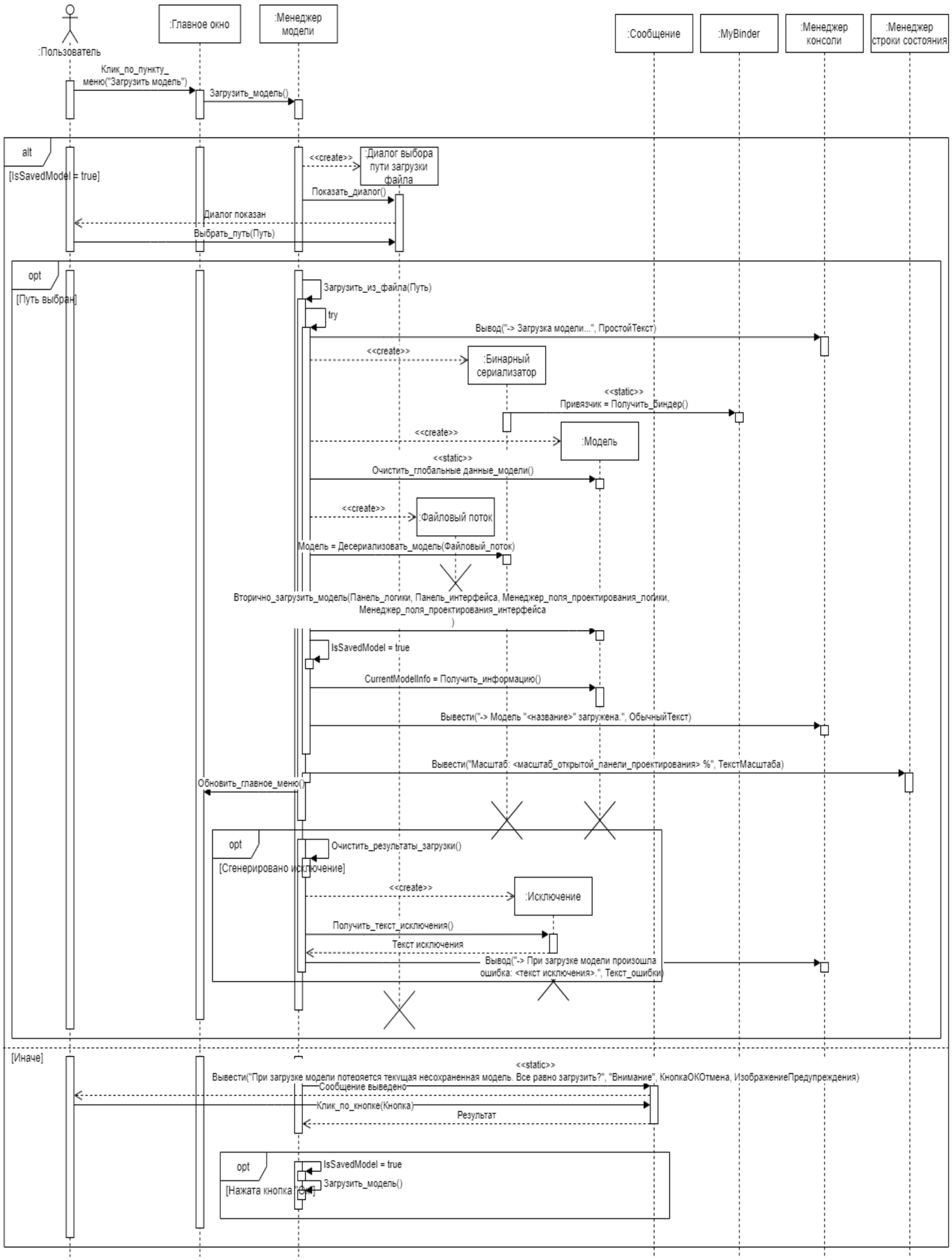
Подп. и дата

Инв. № дубл.

Взам. инв. №

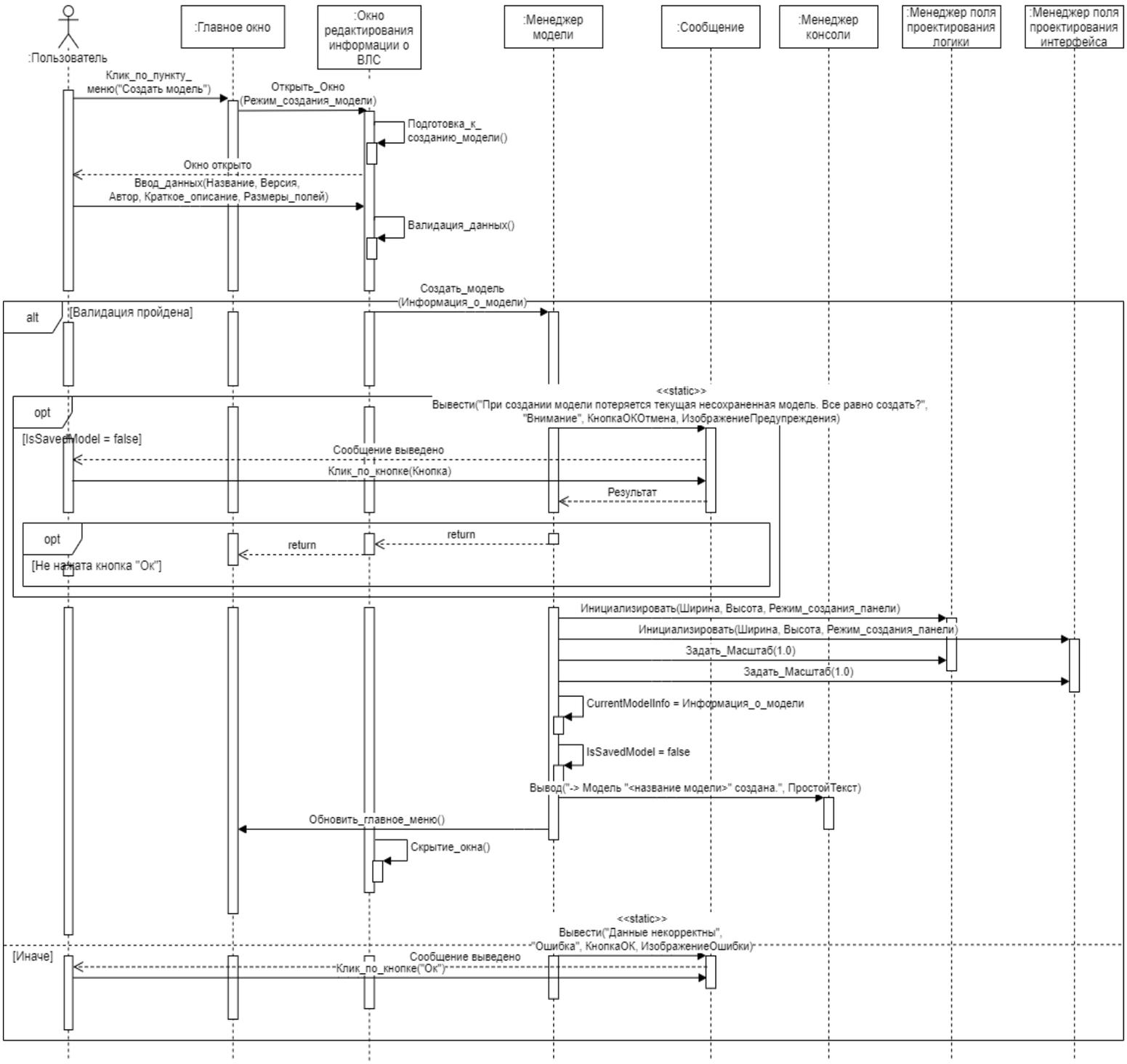
Подп. и дата

Инв. № подл.



Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Диаграмма «Загрузить модель»	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 107	Листов 150
Т.контр.		Ланских Ю.В.					
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Утв.		Ланских Ю.В.					

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.



					ТПЖА 090302.036 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Дождиков И.С.			Диаграмма «Создать модель»	Лит. / Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 108 / Листов 150
Т.контр.		Ланских Ю.В.				
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00
Утв.		Ланских Ю.В.				

Перв. применение

Справ. №

Подп. и дата

Инв. № дубл.

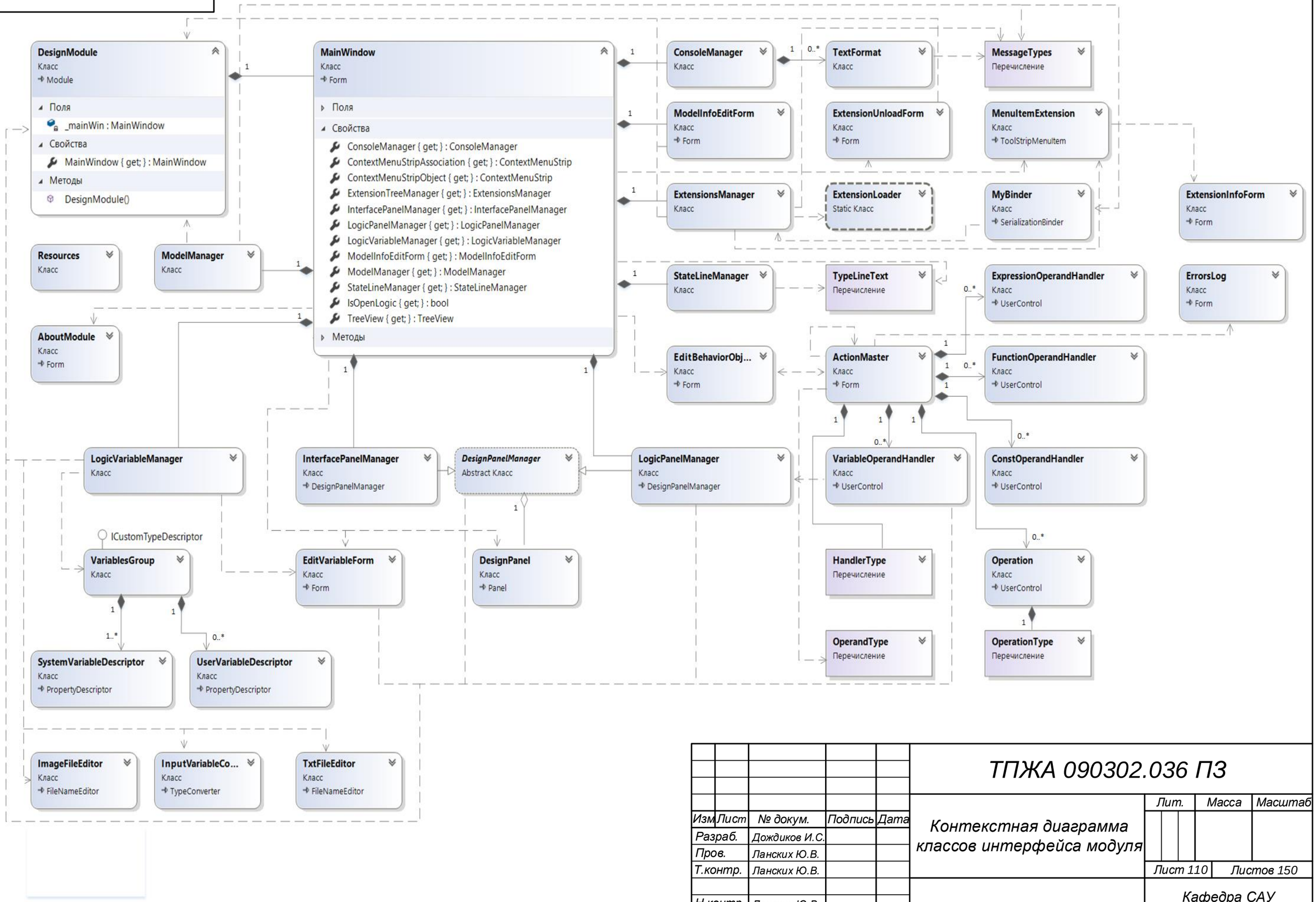
Взам. инв. №

Подп. и дата

Инв. № подл.

Приложение И
(обязательное)
Диаграммы классов

ТПЖА 090302.036 ПЗ



Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Дождиков И.С.		
Пров.		Ланских Ю.В.		
Т.контр.		Ланских Ю.В.		
Н.контр.		Ланских Ю.В.		
Утв.		Ланских Ю.В.		

ТПЖА 090302.036 ПЗ

Контекстная диаграмма классов интерфейса модуля

Лит.	Масса	Масштаб

Лист 110 | Листов 150

Кафедра САУ
Группа ИТб-4301-01-00

Перв. применение | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

MainWindow
Класс
Form

- Поля
- Свойства
  - ConsoleManager { get; } : ConsoleManager
  - ContextMenuStripAssociation { get; } : ContextMenuStrip
  - ContextMenuStripObject { get; } : ContextMenuStrip
  - ExtensionTreeManager { get; } : ExtensionsManager
  - InterfacePanelManager { get; } : InterfacePanelManager
  - IsOpenLogic { get; } : bool
  - LogicPanelManager { get; } : LogicPanelManager
  - LogicVariableManager { get; } : LogicVariableManager
  - ModelInfoEditForm { get; } : ModelInfoEditForm
  - ModelManager { get; } : ModelManager
  - StateLineManager { get; } : StateLineManager
  - TreeView { get; } : TreeView
- Методы
  - addProperty_Click(object sender, EventArgs e) : void
  - contextMenuStripObject_Opening(object sender, CancelEventArgs e) : void
  - contextMenuStripObjectInterface_Opening(object sender, CancelEventArgs e) : void
  - Dispose(bool disposing) : void
  - InitializeComponent() : void
  - InterfacePage_Enter(object sender, EventArgs e) : void
  - InterfacePanel_DragDrop(object sender, DragEventArgs e) : void
  - InterfacePanel_DragEnter(object sender, DragEventArgs e) : void
  - LogicPage_Enter(object sender, EventArgs e) : void
  - LogicPanel_DragDrop(object sender, DragEventArgs e) : void
  - LogicPanel_DragEnter(object sender, DragEventArgs e) : void
  - MainWindow()
  - MainWindow_FormClosed(object sender, FormClosedEventArgs e) : void
  - MainWindow_FormClosing(object sender, FormClosingEventArgs e) : void
  - MainWindow_Shown(object sender, EventArgs e) : void
  - splitContainer2_Panel1_Resize(object sender, EventArgs e) : void
  - toolStripMenuItem3_Click(object sender, EventArgs e) : void
  - toolStripMenuItem4_Click(object sender, EventArgs e) : void
  - toolStripMenuItemAssociation_Click(object sender, EventArgs e) : void
  - toolStripMenuItemDelete_Click(object sender, EventArgs e) : void
  - toolStripMenuItemObjectBehaviour_Click(object sender, EventArgs e) : void
  - TreeViewExtensions_MouseDown(object sender, MouseEventArgs e) : void
  - treeViewExtensions_NodeMouseHover(object sender, TreeNodeMouseHoverEventArgs e) : void
  - UpdateModelMenu() : void
  - настройкиПроектаToolStripMenuItem_Click(object sender, EventArgs e) : void
  - обПрограммеProjectCADToolStripMenuItem_Click(object sender, EventArgs e) : void
  - ОтключитьРасширениеToolStripMenuItem_Click(object sender, EventArgs e) : void
  - открытьПроектToolStripMenuItem_Click(object sender, EventArgs e) : void
  - ПодключитьРасширениеToolStripMenuItem_Click(object sender, EventArgs e) : void
  - создатьМодельToolStripMenuItem_Click(object sender, EventArgs e) : void
  - сохранитьПроектToolStripMenuItem_Click(object sender, EventArgs e) : void

AboutModule
Класс
Form

- Поля
  - components : IContainer
  - label1 : Label
  - label2 : Label
  - label3 : Label
  - label4 : Label
  - pictureBox1 : PictureBox
- Методы
  - AboutModule()
  - Dispose(bool disposing) : void
  - InitializeComponent() : void

Resources
Класс

- Поля
  - resourceCulture : CultureInfo
  - resourceMan : ResourceManager
- Свойства
  - Culture { get; set; } : CultureInfo
  - ResourceManager { get; } : ResourceManager
- Методы
  - Resources()

ConsoleManager
Класс

- Поля
  - _richTextBox : RichTextBox
  - textFormats : List<TextFormat>
- Свойства
  - TextBox { get; } : RichTextBox
- Методы
  - AddLine(string text, MessageTypes type) : void
  - Clear() : void
  - ConsoleManager(RichTextBox richTextBox)
  - Update() : void

TextFormat
Класс

- Свойства
  - Color { get; set; } : Color
  - Length { get; set; } : int
  - Start { get; set; } : int
- Методы
  - TextFormat()
  - TextFormat(int start, int len, Color color)

StateLineManager
Класс

- Поля
  - stateLine : StatusStrip
- Методы
  - SetText(string text, TypeLineText textType) : void
  - StateLineManager(StatusStrip statusStrip)

TypeLineText
Перечисление

- Info
- Scale

MessageTypes
Перечисление

- Simple
- Information
- Warning
- Error

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов интерфейса модуля (часть 1)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 111	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

EditBehaviorObjectForm
Класс
Form

Поля
- buttonAdd : Button
- buttonAddIn : Button
- buttonCancel : Button
- buttonClear : Button
- buttonDelete : Button
- buttonDown : Button
- buttonEdit : Button
- buttonSave : Button
- buttonUp : Button
- components : IContainer
- contextMenuStripAddConstruction : ContextMenuStrip
- imageList1 : ImageList
- syntaxTreeView : TreeView
- toolStripSeparator1 : ToolStripSeparator
- toolStripSeparator2 : ToolStripSeparator
- toolTip1 : ToolTip
- вызовФункцииToolStripMenuItem : ToolStripMenuItem
- подусловиеИЛИЕСЛИToolStripMenuItem : ToolStripMenuItem
- подусловиеИНАЧЕToolStripMenuItem : ToolStripMenuItem
- присвоениеЗначенияToolStripMenuItem : ToolStripMenuItem
- условиеЕСЛИToolStripMenuItem : ToolStripMenuItem
- циклПОКАToolStripMenuItem : ToolStripMenuItem

Свойства
- EventBlocks { get; set; } : Dictionary<EventBlockAttribute, Program>
- IsChanged { get; set; } : bool
- IsChild { get; set; } : bool
- ObjectCAD { get; set; } : ObjectCAD

Методы
- AddChilds(TreeNode treeNode, ExecutionElement element) : void
- AddProgram(TreeNode treeNode, Program program) : void
- buttonAdd_Click(object sender, EventArgs e) : void
- buttonAddIn_Click(object sender, EventArgs e) : void
- buttonCancel_Click(object sender, EventArgs e) : void
- buttonClear_Click(object sender, EventArgs e) : void
- buttonDelete_Click(object sender, EventArgs e) : void
- buttonDown_Click(object sender, EventArgs e) : void
- buttonEdit_Click(object sender, EventArgs e) : void
- buttonSave_Click(object sender, EventArgs e) : void
- buttonUp_Click(object sender, EventArgs e) : void
- CheckBehaviourForUsedObjects() : void
- CheckButtons() : void
- CheckObjects(ExecutionElement executionElement) : void
- Clear() : void
- contextMenuStripAddConstruction_Closed(object sender, ToolStripDropDownClosedEventArgs e) : void
- contextMenuStripAddConstruction_Opening(object sender, CancelEventArgs e) : void
- Dispose(bool disposing) : void
- EditBehaviorObjectForm(ObjectCAD objectCAD)
- EditBehaviorObjectForm_FormClosing(object sender, FormClosingEventArgs e) : void
- GetProgram(TreeNode treeNode) : Program
- GetStructure(TreeNode treeNode) : Structure
- InitializeComponent() : void
- LoadTree() : void
- Save() : void
- SyntaxTreeView_AfterSelect(object sender, TreeViewEventArgs e) : void
- вызовФункцииToolStripMenuItem_Click(object sender, EventArgs e) : void
- подусловиеИЛИЕСЛИToolStripMenuItem_Click(object sender, EventArgs e) : void
- подусловиеИНАЧЕToolStripMenuItem_Click(object sender, EventArgs e) : void
- присвоениеЗначенияToolStripMenuItem_Click(object sender, EventArgs e) : void
- условиеЕСЛИToolStripMenuItem_Click(object sender, EventArgs e) : void
- циклПОКАToolStripMenuItem_Click(object sender, EventArgs e) : void

ConstOperandHandler
Класс
UserControl

Поля

Свойства
- ActionMaster { get; set; } : ActionMaster
- DataVariableType { get; set; } : DataVariableType
- ObjectCAD { get; set; } : ObjectCAD

Методы
- buttonClose_Click(object sender, EventArgs e) : void
- buttonEdit_Click(object sender, EventArgs e) : void
- buttonOpenFile_Click(object sender, EventArgs e) : void
- comboBoxObject_SelectedIndexChanged(object sender, EventArgs e) : void
- ConstOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, bool isRemove)
- ConstOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, ConstOperandExpression constOperandExpression, bool isRemove)
- Dispose(bool disposing) : void
- GetConstOperandExpression() : ConstOperandExpression
- InitializeComponent() : void
- SetType(DataVariableType dataVariableType) : void
- SetValue(ConstOperandExpression constOperandExpression) : void
- выражениеToolStripMenuItem_Click(object sender, EventArgs e) : void
- переменнаяToolStripMenuItem_Click(object sender, EventArgs e) : void
- функцияToolStripMenuItem_Click(object sender, EventArgs e) : void

ExpressionOperandHandler
Класс
UserControl

Поля

Свойства
- ActionMaster { get; set; } : ActionMaster
- Expression { get; set; } : Expression
- HandlerType { get; set; } : HandlerType
- ObjectCAD { get; set; } : ObjectCAD

Методы
- button2_Click(object sender, EventArgs e) : void
- buttonClose_Click(object sender, EventArgs e) : void
- buttonEdit_Click(object sender, EventArgs e) : void
- Dispose(bool disposing) : void
- ExpressionOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, Expression expression, bool isRemove)
- ExpressionOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, HandlerType handlerType, bool isRemove)
- Fill() : void
- GetExecutionElement() : Expression
- InitializeComponent() : void
- label3_MouseEnter(object sender, EventArgs e) : void
- UpdateText() : void
- константаToolStripMenuItem_Click(object sender, EventArgs e) : void
- переменнаяToolStripMenuItem_Click(object sender, EventArgs e) : void
- функцияToolStripMenuItem_Click(object sender, EventArgs e) : void

ErrorsLog
Класс
Form

Поля
- button1 : Button
- columnHeader1 : ColumnHeader
- columnHeader2 : ColumnHeader
- columnHeader3 : ColumnHeader
- components : IContainer
- label1 : Label
- listView1 : ListView

Методы
- button1_Click(object sender, EventArgs e) : void
- Dispose(bool disposing) : void
- ErrorsLog(List<string[]> errorsData)
- InitializeComponent() : void

Operation
Класс
UserControl

Поля

Свойства
- OperationType { get; set; } : OperationType

Методы
- Dispose(bool disposing) : void
- GetValue() : string
- InitializeComponent() : void
- Operation(OperationType operationType)
- SetValue(object value) : void

OperationType
Перечисление
- Setting
- Logic
- Arithmetic

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

					ТПЖА 090302.036 ПЗ		
Изм	Лист	№ докум.	Подпись	Дата	Содержимое классов интерфейса модуля (часть 3)	Лит.	Масса / Масштаб
Разраб.		Дождиков И.С.					
Пров.		Ланских Ю.В.				Лист 113	Листов 150
Т.контр.		Ланских Ю.В.					
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Утв.		Ланских Ю.В.					

ModelManager
Класс

- Свойства
  - CurrentModelInfo { get; set; } : ModelInfo
  - IsSavedModel { get; set; } : bool
- Методы
  - CreateModel(ModelInfo info) : void
  - LoadFromFile(string filePath) : void
  - LoadModel() : void
  - PackModel() : Model
  - SaveIntoFile(string filePath) : void
  - SaveModel() : void
  - SetModelInfo(ModelInfo info) : void

ExtensionsManager
Класс

- Поля
  - extensions : List<Extension>
  - loadedExtensionsMenuItem : ToolStripMenuItem
  - treeView : TreeView
- Свойства
  - Current { get; set; } : ExtensionsManager
  - Extensions { get; } : IReadOnlyCollection<Extension>
- Методы
  - ExtensionsManager(ToolStripMenuItem loadedExtensionsMenuItem, TreeView treeView)
  - Load() : void
  - LoadDefaultExtensions() : void
  - Unload(Guid Id) : void
  - UnloadAll(List<Guid> Id) : void
  - Update() : void

ModelInfoEditForm
Класс
Form

- Поля
  - button1 : Button
  - components : IContainer
  - isCreate : bool
  - label1 : Label
  - label10 : Label
  - label11 : Label
  - label2 : Label
  - label3 : Label
  - label4 : Label
  - label5 : Label
  - label6 : Label
  - label7 : Label
  - label8 : Label
  - label9 : Label
  - maskedTextBox1 : MaskedTextBox
  - maskedTextBox2 : MaskedTextBox
  - maskedTextBox3 : MaskedTextBox
  - numericUpDown1 : NumericUpDown
  - numericUpDown2 : NumericUpDown
  - numericUpDown3 : NumericUpDown
  - numericUpDown4 : NumericUpDown
  - richTextBox1 : RichTextBox
  - tabControl1 : TabControl
  - tabPage1 : TabPage
  - tabPage2 : TabPage
- Методы
  - button1_Click(object sender, EventArgs e) : void
  - DataValidate() : bool
  - Dispose(bool disposing) : void
  - InitializeComponent() : void
  - ModelInfoEditForm()
  - ModelInfoEditForm_FormClosing(object sender, FormClosingEventArgs e) : void
  - Open(bool isCreate) : void

MyBinder
Класс
SerializationBinder

- Поля
  - _changedTypes : IEnumerable<Type>
  - binder : MyBinder
- Методы
  - BindToType(string assemblyName, string typeName) : Type
  - CorrectTypeName(string name) : string
  - GetMyBinder() : MyBinder
  - LoadTypeFromAssembly(string assemblyName, string typeName) : Type
  - MyBinder()
  - MyBinder()

ExtensionLoader
Static Класс

- Поля
  - assemblies : List<Assembly>
- Методы
  - GetClass(XmlNode node, string path) : List<KeyValuePair<string, Type>>
  - Load(string path) : Extension

ExtensionInfoForm
Класс
Form

- Поля
  - components : IContainer
  - label1 : Label
  - label2 : Label
  - label3 : Label
  - label4 : Label
  - richTextBox1 : RichTextBox
  - textBox1 : TextBox
  - textBox2 : TextBox
  - textBox3 : TextBox
- Методы
  - Dispose(bool disposing) : void
  - ExtensionInfoForm()
  - ExtensionInfoForm(Extension extension)
  - InitializeComponent() : void

ExtensionUnloadForm
Класс
Form

- Поля
  - button1 : Button
  - checkedListBox1 : CheckedListBox
  - components : IContainer
  - label1 : Label
- Методы
  - Button1_Click(object sender, EventArgs e) : void
  - Dispose(bool disposing) : void
  - ExtensionUnloadForm()
  - InitializeComponent() : void
  - UpdateList() : void

MenuItemExtension
Класс
ToolStripMenuItem

- Поля
  - extension : Extension
- Методы
  - MenuItemExtension()
  - MenuItemExtension(Extension extension)
  - OnClick(EventArgs e) : void

FunctionOperandHandler
Класс
UserControl

Поля

Свойства
- ActionMaster { get; set; } : ActionMaster
- ButtonArgs { get; set; } : List<Button>
- Current { get; set; } : ObjectCAD
- CurrentObject { get; set; } : ObjectCAD
- LabelArgs { get; set; } : List<Label>
- Method { get; set; } : Method
- Methods { get; set; } : List<Method>
- ObjectCAD { get; set; } : ObjectCAD
- Objects { get; set; } : List<ObjectCAD>
- TextBoxArgs { get; set; } : List<TextBox>

Методы
- buttonArg1_Click(object sender, EventArgs e) : void
- buttonArg2_Click(object sender, EventArgs e) : void
- buttonArg3_Click(object sender, EventArgs e) : void
- buttonClose_Click(object sender, EventArgs e) : void
- buttonEdit_Click(object sender, EventArgs e) : void
- comboBoxFunction_SelectedIndexChanged(object sender, EventArgs e) : void
- comboBoxObject_SelectedIndexChanged(object sender, EventArgs e) : void
- Dispose(bool disposing) : void
- Fill(FunctionOperandExpression functionOperandExpression) : void
- FillArguments(FunctionOperandExpression functionOperandExpression) : void
- FillFunctions(ObjectCAD objectCAD) : void
- FillObjects() : void
- FunctionOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, bool isChangeFunction, bool isRemove)
- FunctionOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, FunctionOperandExpression functionOperand, bool isChangeFunction, bool isRemove)
- GetFunctionOperandExpression() : FunctionOperandExpression
- InitializeComponent() : void
- InitLists() : void
- ShowActionMaster(int index) : void
- textBoxArg1_MouseEnter(object sender, EventArgs e) : void
- textBoxArg2_MouseEnter(object sender, EventArgs e) : void
- textBoxArg3_MouseEnter(object sender, EventArgs e) : void
- выражениеToolStripMenuItem_Click(object sender, EventArgs e) : void
- константаToolStripMenuItem_Click(object sender, EventArgs e) : void
- переменнаяToolStripMenuItem_Click(object sender, EventArgs e) : void

VariableOperandHandler
Класс
UserControl

Поля

Свойства
- ActionMaster { get; set; } : ActionMaster
- InterfaceVariables { get; set; } : Dictionary<string, PropertyInfo>
- ObjectCAD { get; set; } : ObjectCAD
- Objects { get; set; } : List<ObjectCAD>
- operandExpression { get; set; } : VariableOperandExpression
- SystemLogicVariables { get; set; } : Dictionary<string, Tuple<object, PropertyInfo>>
- UserLogicVariables { get; set; } : List<Variable>

Методы
- buttonClose_Click(object sender, EventArgs e) : void
- buttonEdit_Click(object sender, EventArgs e) : void
- comboBoxObject_SelectedIndexChanged(object sender, EventArgs e) : void
- Dispose(bool disposing) : void
- Fill() : void
- FillObjects() : void
- FillVariables(ObjectCAD objectCAD) : void
- GetDataObjects() : List<ObjectCAD>
- GetInterfaceVariables(ObjectCAD objectCAD) : Dictionary<string, PropertyInfo>
- GetVariableOperand() : VariableOperandExpression
- InitializeComponent() : void
- isFileNameValid(string fileName) : bool
- VariableOperandHandler(ObjectCAD objectCAD, ActionMaster actionMaster, bool isRemove, bool isChangeFunction)
- VariableOperandHandler(VariableOperandExpression expression, ObjectCAD objectCAD, ActionMaster actionMaster, bool isRemove, bool isChangeFunction)
- выражениеToolStripMenuItem_Click(object sender, EventArgs e) : void
- константаToolStripMenuItem_Click(object sender, EventArgs e) : void
- функцияToolStripMenuItem_Click(object sender, EventArgs e) : void

InterfacePanelManager
Класс
DesignPanelManager

- Поля
  - propertyTable : PropertyGrid
- Свойства
  - Scale { get; set; } : double
- Методы
  - AddObject(ObjectCAD obj) : void
  - BoxSelected(ElementBox box) : void
  - BoxUnselected(ElementBox box) : void
  - FindObject(ElementBox box) : ObjectCAD
  - InterfacePanelManager(DesignPanel panel, PropertyGrid propertyGrid)
  - Panel_MouseDown(object sender, MouseEventArgs e) : void
  - RemoveObject(ObjectCAD obj) : void
  - SelectObject(ObjectCAD obj) : void
  - UnselectObject() : void

LogicPanelManager
Класс
DesignPanelManager

- Поля
  - firstObject : ObjectCAD
  - selectedLine : Line
- Свойства
  - IsAssociationBuilding { get; set; } : bool
  - Scale { get; set; } : double
  - SelectedLine { get; } : Line
- Методы
  - AddObject(ObjectCAD obj) : void
  - BoxSelected(ElementBox box) : void
  - BoxUnselected(ElementBox box) : void
  - Connect(ObjectCAD @object) : void
  - Connect(ObjectCAD object1, ObjectCAD object2) : void
  - FindObject(ElementBox box) : ObjectCAD
  - LineSelected(Line line) : void
  - LogicPanelManager(DesignPanel panel)
  - Panel_MouseDown(object sender, MouseEventArgs e) : void
  - RemoveObject(ObjectCAD obj) : void
  - SelectObject(ObjectCAD obj) : void
  - UnselectLine() : void
  - UnselectObject() : void

DesignPanelManager
Abstract Класс

- Поля
  - COEFF : int
  - handCursor : Cursor
  - isPressed : bool
  - lastMousePos : Point
  - objects : List<ObjectCAD>
  - scale : double
  - selectedObject : ObjectCAD
- Свойства
  - Height { get; set; } : int
  - Objects { get; } : IReadOnlyCollection<ObjectCAD>
  - Panel { get; set; } : DesignPanel
  - Scale { get; set; } : double
  - SelectedObject { get; } : ObjectCAD
  - Width { get; set; } : int
- Методы
  - *AddObject(ObjectCAD obj) : void*
  - ClearAll() : void
  - DesignPanelManager(DesignPanel panel)
  - Draw() : void
  - *FindObject(ElementBox box) : ObjectCAD*
  - GetPhysicalPoint(Point virtualCoord) : Point
  - GetVirtualCoords(Point physicalCoord) : Point
  - Initialize(int width, int height, bool isNew) : void
  - Panel_MouseDown(object sender, MouseEventArgs e) : void
  - Panel_MouseMove(object sender, MouseEventArgs e) : void
  - Panel_MouseUp(object sender, MouseEventArgs e) : void
  - Panel_MouseWheel(object sender, MouseEventArgs e) : void
  - *RemoveObject(ObjectCAD obj) : void*
  - *SelectObject(ObjectCAD obj) : void*
  - *UnselectObject() : void*
- Вложенные типы

DesignPanel
Класс
Panel

- Методы
  - DesignPanel()

					ТПЖА 090302.036 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата	Содержимое классов интерфейса модуля (часть 6)	Лит. / Масса / Масштаб
Разраб.		Дождиков И.С.				
Пров.		Ланских Ю.В.				Лист 116 / Листов 150
Т.контр.		Ланских Ю.В.				
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00
Утв.		Ланских Ю.В.				

LogicVariableManager
Класс

- Поля
  - AddProperty : Button
  - ChangeButton : ToolStripMenuItem
  - ContextMenuStrip : ContextMenuStrip
  - DeleteButton : ToolStripMenuItem
- Свойства
  - SelectedObject { get; set; } : ObjectCAD
  - Table { get; set; } : PropertyGrid
- Методы
  - ChangeButton_Click(object sender, EventArgs e) : void
  - Clear() : void
  - ContextMenuStrip_Opened(object sender, EventArgs e) : void
  - DeleteButton_Click(object sender, EventArgs e) : void
  - LogicVariableManager(PropertyGrid table, Button addProperty, ContextMenuStrip contextMenuStrip, ToolStripMenuItem changeButton, ToolStripMenuItem deleteButton)
  - SetObject(ObjectCAD @object) : void
  - Update() : void

EditVariableForm
Класс
→ Form

- Поля
  - button1 : Button
  - button2 : Button
  - comboBox1 : ComboBox
  - comboBox2 : ComboBox
  - components : IContainer
  - isCreate : bool
  - label1 : Label
  - label2 : Label
  - label3 : Label
  - label4 : Label
  - label6 : Label
  - textBox1 : TextBox
  - textBox3 : TextBox
  - Variable : Variable
- Методы
  - button1_Click(object sender, EventArgs e) : void
  - button2_Click(object sender, EventArgs e) : void
  - Dispose(bool disposing) : void
  - EditVariableForm()
  - GetDataDirection(string value) : DataDirectionType
  - GetDataType(string value) : DataVariableType
  - InitializeComponent() : void
  - OpenForCreate(PrimitiveObjectType objectType) : void
  - OpenForEdit(Variable variable, PrimitiveObjectType objectType) : void
  - ValidateForm() : bool

ICustomTypeDescriptor

VariablesGroup
Класс

- Поля
  - head : Head
  - input : Input
  - output : Output
  - Owners : Dictionary<PropertyDescriptor, object>
  - Variables : List<Variable>
- Методы
  - GetAttributes() : AttributeCollection
  - GetClassName() : string
  - GetComponentName() : string
  - GetConverter() : TypeConverter
  - GetDefaultEvent() : EventDescriptor
  - GetDefaultProperty() : PropertyDescriptor
  - GetEditor(Type editorBaseType) : object
  - GetEvents() : EventDescriptorCollection
  - GetEvents(Attribute[] attributes) : EventDescriptorCollection
  - GetProperties() : PropertyDescriptorCollection
  - GetProperties(Attribute[] attributes) : PropertyDescriptorCollection
  - GetPropertyOwner(PropertyDescriptor pd) : object
  - VariablesGroup(ObjectCAD obj)

InputVariableConverter
Класс
→ TypeConverter

- Методы
  - GetStandardValues(ITypeDescriptorContext context) : StandardValuesCollection
  - GetStandardValuesExclusive(ITypeDescriptorContext context) : bool
  - GetStandardValuesSupported(ITypeDescriptorContext context) : bool

SystemVariableDescriptor
Класс
→ PropertyDescriptor

- Поля
  - isReadOnly : bool
  - Variable : Variable
- Свойства
  - ComponentType { get; } : Type
  - IsReadOnly { get; } : bool
  - PropertyType { get; } : Type
- Методы
  - CanResetValue(object component) : bool
  - GetValue(object component) : object
  - ResetValue(object component) : void
  - SetValue(object component, object value) : void
  - ShouldSerializeValue(object component) : bool
  - SystemVariableDescriptor(Variable target, Attribute[] attrs, bool IsReadOnly)

UserVariableDescriptor
Класс
→ PropertyDescriptor

- Поля
  - Variable : Variable
- Свойства
  - ComponentType { get; } : Type
  - IsReadOnly { get; } : bool
  - PropertyType { get; } : Type
- Методы
  - CanResetValue(object component) : bool
  - GetValue(object component) : object
  - ResetValue(object component) : void
  - SetValue(object component, object value) : void
  - ShouldSerializeValue(object component) : bool
  - UserVariableDescriptor(Variable target, Attribute[] attrs)

ImageFileEditor
Класс
→ FileNameEditor

- Методы
  - InitializeDialog(OpenFileDialog ofd) : void

TxtFileEditor
Класс
→ FileNameEditor

- Методы
  - InitializeDialog(OpenFileDialog ofd) : void

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов интерфейса модуля (часть 7)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 117	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

ActionMaster
Класс
Form

Поля
- button1 : Button
- button7 : Button
- buttonAddOperand : Button
- components : IContainer
- contextMenuStrip1 : ContextMenuStrip
- DELTA_ADD_BUTTON : Point
- DELTA_OPERAND : Point
- DELTA_OPERAND_HANDLER : Point
- ERRORS : string[,]
- panel1 : Panel
- panel2 : Panel
- pictureBox1 : PictureBox
- START_ADD_BUTTON : Point
- START_OPERAND : Point
- toolStripSeparator1 : ToolStripSeparator
- toolStripSeparator2 : ToolStripSeparator
- выражениеToolStripMenuItem : ToolStripMenuItem
- константаToolStripMenuItem : ToolStripMenuItem
- переменнаяToolStripMenuItem : ToolStripMenuItem
- функцияToolStripMenuItem : ToolStripMenuItem

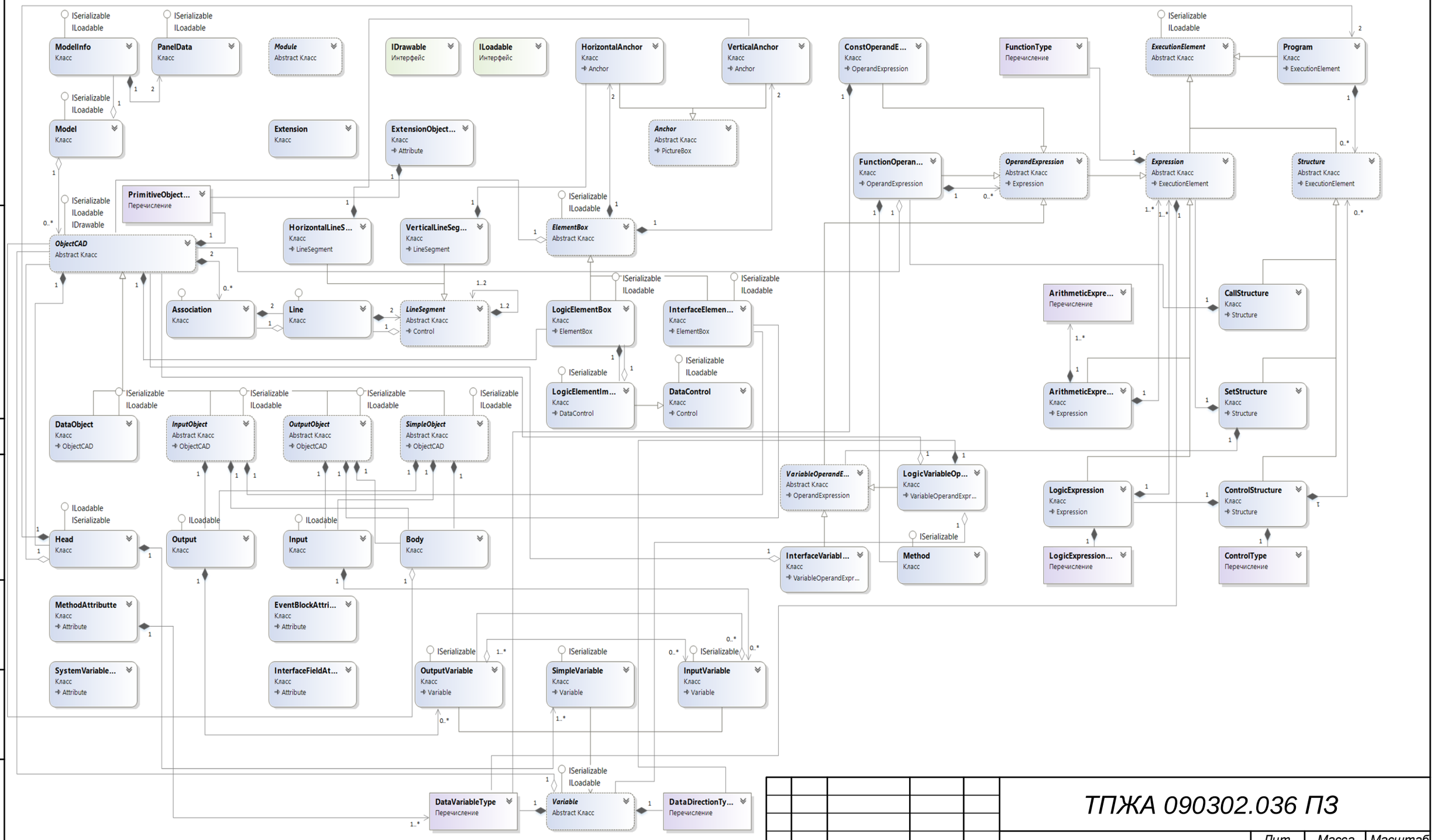
Свойства
- ControlType { get; set; } : ControlType
- DataTypeArgument { get; set; } : DataVariableType
- Elements { get; set; } : List<UserControl>
- ExecutionElement { get; set; } : ExecutionElement
- ExpressionOperandHandler { get; set; } : ExpressionOperandHandler
- HandlerType { get; set; } : HandlerType
- IsChanged { get; set; } : bool
- IsChild { get; set; } : bool
- ObjectArgument { get; set; } : TextBox
- ObjectCAD { get; set; } : ObjectCAD
- OperandsCount { get; set; } : int
- TreeNode { get; set; } : TreeNode

Методы
- ActionMaster(ObjectCAD objectCAD, ControlType controlType, TreeNode treeNode, bool isChild)
- ActionMaster(ObjectCAD objectCAD, ExecutionElement executionElement, TreeNode treeNode)
- ActionMaster(ObjectCAD objectCAD, ExpressionOperandHandler expressionOperandHandler, ExecutionElement executionElement)
- ActionMaster(ObjectCAD objectCAD, HandlerType handlerType, ExpressionOperandHandler expressionOperandHandler)
- ActionMaster(ObjectCAD objectCAD, HandlerType handlerType, TreeNode treeNode, bool isChild)
- ActionMaster(ObjectCAD objectCAD, TextBox objectArgument, DataVariableType dataTypeArgument)
- ActionMaster_FormClosing(object sender, FormClosingEventArgs e) : void
- AddOperand(OperandType operandType) : void
- AddOperation() : void
- button1_Click(object sender, EventArgs e) : void
- button7_Click(object sender, EventArgs e) : void
- buttonAddOperand_Click(object sender, EventArgs e) : void
- ChangeOperand(UserControl from, UserControl to) : void
- ChangeOperandToCall(UserControl from) : void
- ChangeOperandToExpression(UserControl from) : void
- ChangeOperandToVariable(UserControl from) : void
- ChangerOperandToConst(UserControl from) : void
- CheckErrors() : bool
- contextMenuStrip1_Closed(object sender, ToolStripDropDownClosedEventArgs e) : void
- contextMenuStrip1_Opening(object sender, CancelEventArgs e) : void
- Dispose(bool disposing) : void
- GetArithmeticExpressionType(string value) : ArithmeticExpressionType
- GetExecutionElement() : ExecutionElement
- GetExecutionElement(UserControl userControl) : ExecutionElement
- GetLogicExpressionType(string value) : LogicExpressionType
- InitializeComponent() : void
- LoadExpression(Expression expression) : void
- LoadOperand(Expression operand) : void
- newOperand(Expression operand, OperandType operandType) : void
- RemoveOperand(UserControl operand) : void
- выражениеToolStripMenuItem_Click(object sender, EventArgs e) : void
- константаToolStripMenuItem_Click(object sender, EventArgs e) : void
- переменнаяToolStripMenuItem_Click(object sender, EventArgs e) : void
- функцияToolStripMenuItem_Click(object sender, EventArgs e) : void

HandlerType
Перечисление
- Setting
- Logic
- Arithmetic
- Calling
- Argument

OperandType
Перечисление
- Const
- Expression
- Function
- Variable

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов интерфейса модуля (часть 8)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 118	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					



Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Дождиков И.С.		
Пров.		Ланских Ю.В.		
Т.контр.		Ланских Ю.В.		
Н.контр.		Ланских Ю.В.		
Утв.		Ланских Ю.В.		

ТПЖА 090302.036 ПЗ

Контекстная диаграмма классов структур данных

Лит.	Масса	Масштаб

Лист 119 | Листов 150

Кафедра САУ
Группа ИТб-4301-01-00

Перв. применение

Справ. №

Подл. и дата

Инв. № дубл.

Взам. инв. №

Подл. и дата

Инв. № подл.

ObjectCAD
Abstract Класс

- Поля
  - _idList : List<ulong>
  - Objects : Dictionary<ulong, ObjectCAD>
- Свойства
  - Associations { get; set; } : List<Association>
  - ExtensionId { get; set; } : Guid
  - Head { get; set; } : Head
  - Id { get; set; } : ulong
  - Image { get; } : Bitmap
  - IsActive { get; set; } : bool
  - IsBehaviorChange { get; set; } : bool
  - IsVariablesEdit { get; set; } : bool
  - LogicBox { get; set; } : LogicElementBox
  - Name { get; set; } : string
  - PrimitiveType { get; set; } : PrimitiveObjectType
  - UsedObjects { get; set; } : List<ulong>
- Методы
  - ClearIds() : void
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - ObjectCAD(PrimitiveObjectType primitiveType, bool isBehaviorChange, bool isVariablesEdit, Guid extensionId, Control panel, Bitmap image, Point pos, Size size)
  - ObjectCAD(SerializationInfo info, StreamingContext context)
  - RemoveId() : void

SimpleObject
Abstract Класс

- Свойства
  - Body { get; set; } : Body
  - Input { get; set; } : Input
  - Output { get; set; } : Output
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - SimpleObject(bool isBehaviorChange, bool isVariablesEdit, Guid extensionId, Control panel, Bitmap image, Point pos, Size size)
  - SimpleObject(SerializationInfo serializationInfo, StreamingContext streamingContext)

DataObject
Класс

- Методы
  - DataObject(Guid extensionId, Control panel, Bitmap image, Point pos, Size size)
  - DataObject(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void

PrimitiveObject...
Перечисление

- InputObject
- OutputObject
- SimpleObject
- DataObject

InputObject
Abstract Класс

- Свойства
  - Body { get; set; } : Body
  - InterfaceElementBox { get; set; } : InterfaceElementBox
  - Output { get; set; } : Output
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - InputObject(bool isBehaviorChange, bool isVariablesEdit, Guid extensionId, Control logicPanel, Control interfacePanel, Control children, Bitmap image, Point pos, Size size)
  - InputObject(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void

OutputObject
Abstract Класс

- Свойства
  - Body { get; set; } : Body
  - Input { get; set; } : Input
  - InterfaceElementBox { get; set; } : InterfaceElementBox
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - OutputObject(bool isBehaviorChange, bool isVariablesEdit, Guid extensionId, Control logicPanel, Control interfacePanel, Control children, Bitmap image, Point pos, Size size)
  - OutputObject(SerializationInfo serializationInfo, StreamingContext streamingContext)

Head
Класс

- Поля
  - objectId : ulong
  - variables : List<ulong>
- Свойства
  - Name { get; set; } : SimpleVariable
  - ObjectCAD { get; set; } : ObjectCAD
  - StartBlock { get; set; } : Program
  - UpdateBlock { get; set; } : Program
  - Variables { get; set; } : List<SimpleVariable>
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Head(ObjectCAD obj)
  - Head(ObjectCAD obj, List<SimpleVariable> list)
  - Head(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - InitializeSystemVars() : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - SetSystemValue(ulong idSystemVariable) : void

Body
Класс

- Свойства
  - Object { get; set; } : ObjectCAD
- Методы
  - Body(ObjectCAD objectCAD)

Anchor
Abstract Класс
PictureBox

- Поля
  - delta : int[]
  - isPressed : bool
  - lastClip : Rectangle
  - lastPos : Point
  - panel : Control
  - START_HEIGHT : int
  - START_WIDTH : int
  - startMousePos : Point
- Свойства
  - Delta { get; } : int[]
  - IsPressed { get; } : bool
  - LastClip { get; } : Rectangle
  - LastPos { get; set; } : Point
  - Panel { get; } : Control
  - StartMousePos { get; } : Point
- Методы
  - Anchor(Control panel, Point position)
  - OnMouseDown(MouseEventArgs e) : void
  - OnMouseHover(EventArgs e) : void
  - OnMouseLeave(EventArgs e) : void
  - OnMouseUp(MouseEventArgs e) : void
  - RegisterEventMoved(Point startPos, Point endPos) : void
- События
  - AnchorMouseUp : AnchorMouseUpHandler
  - AnchorMoved : AnchorMoveHandler
  - AnchorStopped : AnchorHoverHandler
- Вложенные типы

HorizontalAnchor
Класс

- Методы
  - HorizontalAnchor(Control panel, Point position)
  - OnMouseEnter(EventArgs e) : void
  - OnMouseMove(MouseEventArgs e) : void

VerticalAnchor
Класс

- Методы
  - OnMouseEnter(EventArgs e) : void
  - OnMouseMove(MouseEventArgs e) : void
  - VerticalAnchor(Control panel, Point position)

Output
Класс

- Свойства
  - Variables { get; set; } : List<OutputVariable>
- Методы
  - InitializeSystemVars() : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - Output()
  - Output(List<OutputVariable> list)
  - SetSystemValue(ulong idSystemVariable) : void

Input
Класс

- Свойства
  - Variables { get; set; } : List<InputVariable>
- Методы
  - InitializeSystemVars() : void
  - Input()
  - Input(List<InputVariable> list)
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - SetSystemValue(ulong idSystemVariable) : void

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов структур данных (часть 2)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 121	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

LogicElementBox
Класс

- Поля
  - lastPoint : Point
- Методы
  - Element_MouseDown(object sender, MouseEventArgs e) : void
  - Element_MouseMove(object sender, MouseEventArgs e) : void
  - Element_MouseUp(object sender, MouseEventArgs e) : void
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - LogicElementBox(ObjectCAD obj, Control panel, Control children, Point position, double scaleCoeff)
  - LogicElementBox(SerializationInfo serializationInfo, StreamingContext streamingContext)

InterfaceElementBox
Класс

- Методы
  - Element_MouseMove(object sender, MouseEventArgs e) : void
  - InterfaceElementBox(ObjectCAD obj, Control panel, Control children, Point position, double scaleCoeff)
  - InterfaceElementBox(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void

LogicElementImage
Класс

- Поля
  - borders : PictureBox
  - box : LogicElementBox
  - image : PictureBox
  - scale : double
  - text : Label
- Свойства
  - Bitmap { get; } : PictureBox
  - Box { get; set; } : LogicElementBox
  - ObjectText { get; } : Label
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - LogicElementImage(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - LogicElementImage(Size size, string text, Bitmap image)
  - OnResize(EventArgs eventargs) : void

DataControl
Класс
→ Control

- Поля
  - location : Point
  - size : Size
- Методы
  - DataControl()
  - DataControl(SerializationInfo info, StreamingContext context)
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void

ElementBox
Abstract Класс

- Поля
  - box : PictureBox
  - children : Control
  - delta : int[]
  - hAnchors : HorizontalAnchor[]
  - isPressed : bool
  - isSelected : bool
  - maxSize : Size
  - minSize : Size
  - objCAD : ObjectCAD
  - originLoc : Point
  - originMaxSize : Size
  - originMinSize : Size
  - originSize : Size
  - panel : Control
  - scaleCoeff : double
  - vAnchors : VerticalAnchor[]
- Свойства
  - Children { get; } : Control
  - HAnchors { get; } : HorizontalAnchor[]
  - IsSelected { get; set; } : bool
  - Object { get; } : ObjectCAD
  - Panel { get; } : Control
  - ScaleCoeff { get; } : double
  - VAnchors { get; } : VerticalAnchor[]
- Методы
  - Element_MouseClick(object sender, MouseEventArgs e) : void
  - Element_MouseDown(object sender, MouseEventArgs e) : void
  - Element_MouseEnter(object sender, EventArgs e) : void
  - Element_MouseLeave(object sender, EventArgs e) : void
  - Element_MouseMove(object sender, MouseEventArgs e) : void
  - Element_MouseUp(object sender, MouseEventArgs e) : void
  - Element_PaintBackground(object sender, PaintEventArgs e) : void
  - ElementBox(ObjectCAD obj, Control panel, Control children, Point position, double scaleCoeff)
  - ElementBox(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - GetNewPos() : Point[]
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - HorizontalAnchorLeft_AnchorMoved(Point lastPos, Point newPos) : void
  - HorizontalAnchorRight_AnchorMoved(Point lastPos, Point newPos) : void
  - InitEvents() : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - RemoveFromPanel() : void
  - Scale(double factor) : void
  - UpdateAnchorsPos() : void
  - VerticalAnchorDown_AnchorMoved(Point lastPos, Point newPos) : void
  - VerticalAnchorUp_AnchorMoved(Point lastPos, Point newPos) : void
- События
  - BoxMoved : ElementBoxHandler
  - BoxSelected : ElementBoxHandler
  - BoxUnselected : ElementBoxHandler
- Вложенные типы

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов структур данных (часть 3)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 122	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

ТПЖА 090302.036 ПЗ

Line
Класс

- Поля
  - association : Association
  - BORDER_WIDTH : int
  - endPoint : Point
  - endSeg : LineSegment
  - isSelected : bool
  - LINE_WIDTH : int
  - originEndPoint : Point
  - originStartPoint : Point
  - panel : Control
  - points : List<Point>
  - scaleCoeff : double
  - startPoint : Point
  - startSeg : LineSegment
  - STEP_LOC : int
- Свойства
  - Association { get; } : Association
  - EndPoint { get; } : Point
  - EndSegment { get; set; } : LineSegment
  - IsSelected { get; set; } : bool
  - Panel { get; set; } : Control
  - ScaleCoeff { get; } : double
  - StartPoint { get; } : Point
  - StartSegment { get; set; } : LineSegment
- Методы
  - CanUse(Point point, Control panel, bool isCollision, Control[] ignoredControls) : bool
  - DrawAll() : void
  - GetCenter(Control control) : Point
  - GetFullPath(Point start, Point end, Control panel, bool isCollision, Control[] ignoredControls) : List<Point>
  - GetLocation(Point cur) : List<Point>
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - GetPath() : List<Point>
  - GetPath(Control start, Control end, Control panel, Control[] ignoredControls) : List<Point>
  - Hide() : void
  - Line(List<Point> path, Control panel, Association association, double scaleCoeff)
  - Line(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - MovedSegmentEvent(LineSegment seg, Point lastPos, Point newPos) : void
  - RemoveFromPanel() : void
  - Scale(double factor) : void
  - SetColor(Color c) : void
  - SetPath(List<Point> newPath) : void
  - SetSegment(LineSegment seg, Point start, Point end) : void
  - Show() : void
- События
  - LineMoved : LineHandler
  - LineSelected : LineHandler
  - LineUnselected : LineHandler
- Вложенные типы

Association
Класс

- Поля
  - Associations : List<Association>
  - Ids : ulong[]
  - line : Line
  - objects : ObjectCAD[]
- Свойства
  - Line { get; } : Line
  - Objects { get; } : ObjectCAD[]
- Методы
  - Association(ObjectCAD[] objects, double scale)
  - Association(SerializationInfo info, StreamingContext context)
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void

LineSegment
Abstract Класс
→ Control

- Поля
  - isHasAnchor : bool
  - MIN_DELTA_ANCHOR : int
  - nearestSegments : LineSegment[]
  - originPoint : Point
  - originSize : Size
  - parent : Line
- Свойства
  - IsHasAnchor { get; } : bool
  - NearestSegments { get; } : LineSegment[]
  - OriginPoint { get; } : Point
  - OriginSize { get; } : Size
  - ParentLine { get; } : Line
- Методы
  - AnchorMouseUp() : void
  - AnchorStopped() : void
  - DrawAll() : void
  - LineSegment(Line parent, LineSegment[] nearestSegs)
  - OnClick(EventArgs e) : void
  - OnMouseEnter(EventArgs e) : void
  - OnMouseLeave(EventArgs e) : void
  - OnPaint(PaintEventArgs e) : void
  - PutSegment(ref List<object> segments) : void
  - *RemoveFromPanel() : void*
  - *ScaleSegment(double factor) : void*
  - SetColorAll(Color c) : void
  - *SetLocationAndSize(Point p1, Point p2, bool isInit) : void*
  - SetPanel(Control panel) : void
  - *SetSelected(bool isSelect) : void*
  - SetVisible(bool isShow) : void
  - *UpdateAnchorPos() : void*
  - *UpdateSize() : void*

HorizontalLineSegment
Класс

- Поля
  - horizontalAnchor : VerticalAnchor
- Свойства
  - HorizontalAnchor { get; } : VerticalAnchor
- Методы
  - HorizontalAnchorMoved(Point lastPos, Point newPos) : void
  - HorizontalLineSegment(Point startPos, Point endPos, Line parent, Control panel, LineSegment[] nearestSegs, bool isHasAnchor)
  - OnPaint(PaintEventArgs e) : void
  - RemoveFromPanel() : void
  - ScaleSegment(double factor) : void
  - SetLocationAndSize(Point p1, Point p2, bool isInit) : void
  - SetSelected(bool isSelect) : void
  - UpdateAnchorPos() : void
  - UpdateSize() : void

VerticalLineSegment
Класс

- Поля
  - verticalAnchor : HorizontalAnchor
- Свойства
  - VerticalAnchor { get; } : HorizontalAnchor
- Методы
  - OnPaint(PaintEventArgs e) : void
  - RemoveFromPanel() : void
  - ScaleSegment(double factor) : void
  - SetLocationAndSize(Point p1, Point p2, bool isInit) : void
  - SetSelected(bool isSelect) : void
  - UpdateAnchorPos() : void
  - UpdateSize() : void
  - VerticalAnchorMoved(Point lastPos, Point newPos) : void
  - VerticalLineSegment(Point startPos, Point endPos, Line parent, Control panel, LineSegment[] nearestSegs, bool isHasAnchor)

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов структур данных (часть 4)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 123	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

OutputVariable
Класс

- Поля
  - InputVariablesId : ulong[]
- Свойства
  - Data { get; set; } : object
  - InputVariables { get; set; } : List<InputVariable>
- Методы
  - DisconnectForObject(ObjectCAD objectCAD) : void
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - GetOutputVariableByRef(string variableRef, ObjectCAD objectCAD) : OutputVariable
  - GetOutputVariablesFromRef(DataVariableType dataVariableType, ObjectCAD selectedObject) : List<OutputVariable>
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - OutputVariable(ObjectCAD objectCAD, string name, string description, DataVariableType dataType)
  - OutputVariable(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - ResetVariable(Variable var) : void
  - SetVariable(Variable var) : void
  - ToString() : string

DataDirectionType
Перечисление

- In
- Out
- Simple

DataVariableType
Перечисление

- Int
- Double
- String
- Bool
- TextFile
- ImageFile

SimpleVariable
Класс

- Методы
  - SimpleVariable(ObjectCAD objectCAD, string name, string description, DataVariableType dataType)
  - SimpleVariable(SerializationInfo serializationInfo, StreamingContext streamingContext)

InputVariable
Класс

- Поля
  - OutputVariableId : ulong
- Свойства
  - OutputVariable { get; set; } : OutputVariable
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - InputVariable(ObjectCAD objectCAD, string name, string description, DataVariableType dataType)
  - InputVariable(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - ResetVariable(Variable var) : void
  - SetVariable(Variable var) : void

Variable
Abstract Класс

- Поля
  - _idList : List<ulong>
  - data : object
  - ObjectId : ulong
  - Variables : List<Variable>
- Свойства
  - Data { get; set; } : object
  - DataDirection { get; set; } : DataDirectionType
  - DataType { get; set; } : DataVariableType
  - Description { get; set; } : string
  - Id { get; set; } : ulong
  - Name { get; set; } : string
  - Object { get; set; } : ObjectCAD
- Методы
  - ClearIds() : void
  - Connect(InputVariable inputVariable, OutputVariable outputVariable) : bool
  - Disconnect(InputVariable inputVariable, OutputVariable outputVariable) : bool
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - RemoveId() : void
  - ResetVariable(Variable var) : void
  - SetVariable(Variable var) : void
  - Variable(ObjectCAD objectCAD, string name, string description, DataDirectionType dataDirection, DataVariableType dataType, object data)
  - Variable(SerializationInfo info, StreamingContext context)

Model
Класс

- Свойства
  - Extensions { get; set; } : Dictionary<Guid, string>
  - IsLoad { get; set; } : bool
  - ModelInfo { get; set; } : ModelInfo
  - Objects { get; set; } : List<object>
  - VersionCAD { get; set; } : string
- Методы
  - Clear() : void
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - Model(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - Model(string versionCAD, ModelInfo modelInfo, List<object> objects)

PanelData
Класс

- Свойства
  - PanelScale { get; set; } : double
  - PanelSize { get; set; } : Size
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - PanelData()
  - PanelData(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - PanelData(Size panelSize, double panelScale)

IDrawable
Интерфейс

- Свойства
  - *Image { get; } : Bitmap*

ILoadable
Интерфейс

- Методы
  - *Load(Panel logicPanel, Panel inte...*

ModelInfo
Класс

- Свойства
  - Author { get; set; } : string
  - Description { get; set; } : string
  - InterfacePanelData { get; set; } : PanelData
  - LogicPanelData { get; set; } : PanelData
  - Name { get; set; } : string
  - Version { get; set; } : string
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - ModelInfo()
  - ModelInfo(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - ModelInfo(string name, string author, string description, string version, PanelData interfacePanelData, PanelData logicPanelData)

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов структур данных (часть 6)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 125	Листов 150
Т.контр.		Ланских Ю.В.					
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Утв.		Ланских Ю.В.					

Перв. применение

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

VariableOperandExpression
Abstract Класс
→ OperandExpression

- Свойства
  - IsReadOnly { get; set; } : bool
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - VariableOperandExpression(FunctionType functionType, bool isReadOnly, DataVariableType expressionType)
  - VariableOperandExpression(SerializationInfo serializationInfo, StreamingContext streamingContext)

LogicVariableOperand
Класс

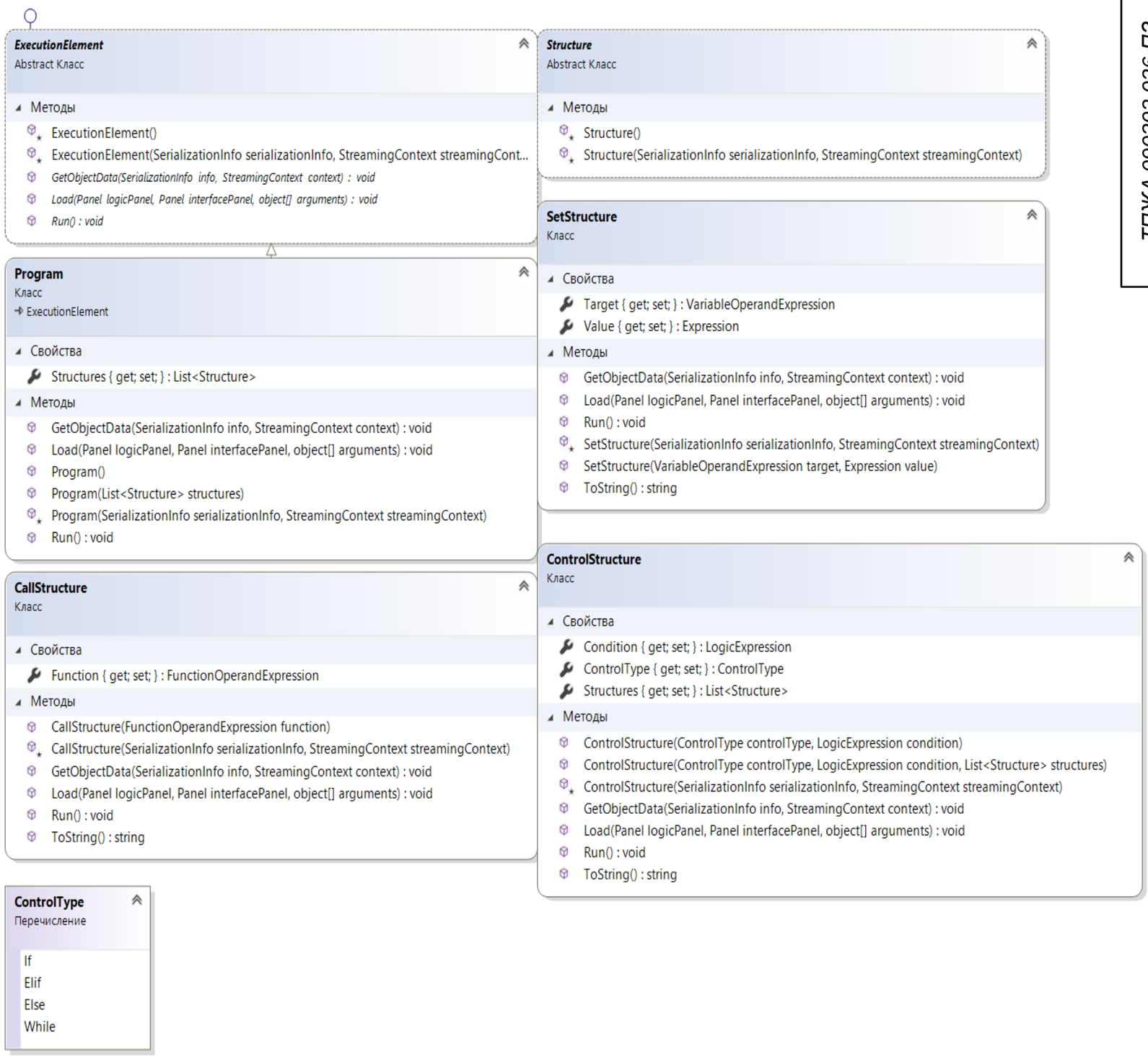
- Поля
  - variableId : ulong
- Свойства
  - DataDirectionType { get; set; } : DataDirectionType
  - objectCAD { get; set; } : ObjectCAD
  - objectId { get; set; } : ulong
  - Variable { get; set; } : Variable
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - LogicVariableOperand(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - LogicVariableOperand(Variable variable, FunctionType functionType, bool isReadOnly, DataVariableType expressionType)
  - Run() : void
  - ToString() : string

InterfaceVariableOperand
Класс

- Поля
  - objectId : ulong
  - propertyName : string
- Свойства
  - ObjectCAD { get; set; } : ObjectCAD
  - Property { get; set; } : PropertyInfo
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - InterfaceVariableOperand(PropertyInfo property, ObjectCAD objectCAD, FunctionType functionType, bool isReadOnly, DataVariableType expressionType)
  - InterfaceVariableOperand(SerializationInfo serializationInfo, StreamingContext streamingContext)
  - Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
  - Run() : void
  - ToString() : string

Method
Класс

- Свойства
  - Arguments { get; set; } : Dictionary<string, DataVariableType>
  - Description { get; set; } : string
  - IsVoid { get; set; } : bool
  - Name { get; set; } : string
  - ReturnedType { get; set; } : DataVariableType
- Методы
  - GetObjectData(SerializationInfo info, StreamingContext context) : void
  - Method()
  - Method(SerializationInfo info, StreamingContext context)
  - Method(string name, string description, DataVariableType returnedType, Dictionary<string, DataVariableType> arguments, bool isVoid)



					ТПЖА 090302.036 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата	Содержимое классов структур данных (часть 8)	Лит. / Масса / Масштаб
Разраб.		Дождиков И.С.				
Пров.		Ланских Ю.В.				Лист 127 / Листов 150
Т.контр.		Ланских Ю.В.				
Н.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00
Утв.		Ланских Ю.В.				

Перв. применение

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Module
Abstract Класс

- Поля
  - author : string
  - description : string
  - name : string
  - version : string
- Свойства
  - Author { get; } : string
  - Current { get; set; } : Module
  - Description { get; } : string
  - Name { get; } : string
  - Version { get; } : string

ExtensionObjectAttribute
Класс
→ Attribute

- Свойства
  - PrimitiveObjectType { get; set; } : PrimitiveObjectType
- Методы
  - ExtensionObjectAttribute()
  - ExtensionObjectAttribute(PrimitiveObjectType primitiveObjectType)

EventBlockAttribute
Класс
→ Attribute

- Свойства
  - Description { get; set; } : string
  - Name { get; set; } : string
- Методы
  - EventBlockAttribute(string name, string description)

Extension
Класс

- Свойства
  - AdditionalTypes { get; set; } : List<Type>
  - Author { get; set; } : string
  - Classes { get; set; } : Dictionary<string, Type>
  - Description { get; set; } : string
  - Name { get; set; } : string
  - UniqueID { get; set; } : Guid
  - Version { get; set; } : string
- Методы
  - Extension(Guid uniqueId, string name, string author, string version, string description, Dictionary<string, Type> classes, List<Type> additionalTypes)

SystemVariableAttribute
Класс
→ Attribute

- Свойства
  - IsReadOnly { get; set; } : bool
- Методы
  - SystemVariableAttribute()
  - SystemVariableAttribute(bool isReadOnly)

MethodAttributte
Класс
→ Attribute

- Свойства
  - ArgNames { get; set; } : string[]
  - ArgTypes { get; set; } : DataVariableType[]
  - Description { get; set; } : string
  - IsVoid { get; set; } : bool
  - Name { get; set; } : string
  - ReturnDataType { get; set; } : DataVariableType
- Методы
  - MethodAttributte()
  - MethodAttributte(string name, string description, string[] argNames, DataVariableType[] argTypes)
  - MethodAttributte(string name, string description, string[] argNames, DataVariableType[] argTypes, DataVariableType returnDataType)

InterfaceFieldAttribute
Класс
→ Attribute

- Свойства
  - Description { get; set; } : string
  - IsReadOnly { get; set; } : bool
  - Name { get; set; } : string
- Методы
  - InterfaceFieldAttribute()
  - InterfaceFieldAttribute(string name, string description, bool isReadOnly)

Изм.	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов структур данных (часть 9)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 128	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

Expression
Abstract Класс
→ ExecutionElement

Свойства
- ExpressionType { get; set; } : DataVariableType
- FunctionType { get; set; } : FunctionType

Методы
- Expression(FunctionType functionType, DataVariableType expressionType)
- Expression(SerializationInfo serializationInfo, StreamingContext streamingContext)
- GetFunctionString(FunctionType functionType) : string
- GetObjectData(SerializationInfo info, StreamingContext context) : void

ArithmeticExpre...
Перечисление

- Addition
- Substraction
- Multiplication
- Division

LogicExpression...
Перечисление

- And
- Or
- Equals
- NotEquals
- More
- Less
- LessOrEquals
- MoreOrEquals

OperandExpression
Abstract Класс
→ Expression

Методы
- GetObjectData(SerializationInfo info, StreamingContext context) : void
- Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
- OperandExpression(FunctionType functionType, DataVariableType expressionType)
- OperandExpression(SerializationInfo serializationInfo, StreamingContext streamingContext)
- Run() : void

ArithmeticExpression
Класс
→ Expression

Свойства
- Operands { get; set; } : List<Expression>
- Operations { get; set; } : List<ArithmeticExpressionType>

Методы
- ArithmeticExpression(List<ArithmeticExpressionType> operations, List<Expression> operands, FunctionType functionType)
- ArithmeticExpression(SerializationInfo serializationInfo, StreamingContext streamingContext)
- GetObjectData(SerializationInfo info, StreamingContext context) : void
- GetStringArithmeticType(ArithmeticExpressionType arithmeticExpressionType) : string
- Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
- Run() : void
- ToString() : string

FunctionType
Перечисление

- None
- Exp
- Pow2
- Sqrt
- Lg
- Ln
- Sin
- Cos
- Tg
- Ctg
- Int
- Double
- String

ConstOperandExpression
Класс

Свойства
- DataConstType { get; set; } : DataVariableType
- Value { get; set; } : object

Методы
- ConstOperandExpression(DataVariableType dataConstType, object value, FunctionType functionType, DataVariableType expressionType)
- ConstOperandExpression(SerializationInfo serializationInfo, StreamingContext streamingContext)
- GetObjectData(SerializationInfo info, StreamingContext context) : void
- GetValue() : string
- Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
- Run() : void
- ToString() : string

LogicExpression
Класс
→ Expression

Свойства
- LogicExpressionType { get; set; } : LogicExpressionType
- Operands { get; set; } : List<Expression>

Методы
- GetObjectData(SerializationInfo info, StreamingContext context) : void
- GetStringLogicType(LogicExpressionType logicExpressionType) : string
- Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
- LogicExpression(LogicExpressionType logicExpressionType, List<Expression> operands, FunctionType functionType)
- LogicExpression(SerializationInfo serializationInfo, StreamingContext streamingContext)
- Run() : void
- ToString() : string

FunctionOperandExpression
Класс

Поля
- objectId : ulong

Свойства
- Arguments { get; set; } : List<OperandExpression>
- Method { get; set; } : Method
- ObjectCAD { get; set; } : ObjectCAD

Методы
- FunctionOperandExpression(Method method, ObjectCAD objectCAD, List<OperandExpression> arguments, FunctionType functionType, DataVariableType expressionType)
- FunctionOperandExpression(SerializationInfo serializationInfo, StreamingContext streamingContext)
- GetObjectData(SerializationInfo info, StreamingContext context) : void
- Load(Panel logicPanel, Panel interfacePanel, object[] arguments) : void
- Run() : void
- ToString() : string

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.036 ПЗ		
Разраб.		Дождиков И.С.			Содержимое классов структур данных (часть 10)	Лит.	Масса / Масштаб
Пров.		Ланских Ю.В.				Лист 129	Листов 150
Т.контр.		Ланских Ю.В.				Кафедра САУ Группа ИТб-4301-01-00	
Н.контр.		Ланских Ю.В.					
Утв.		Ланских Ю.В.					

Перв. применение
Справ. №
Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

Приложение К
(обязательное)
Диаграммы компонентов

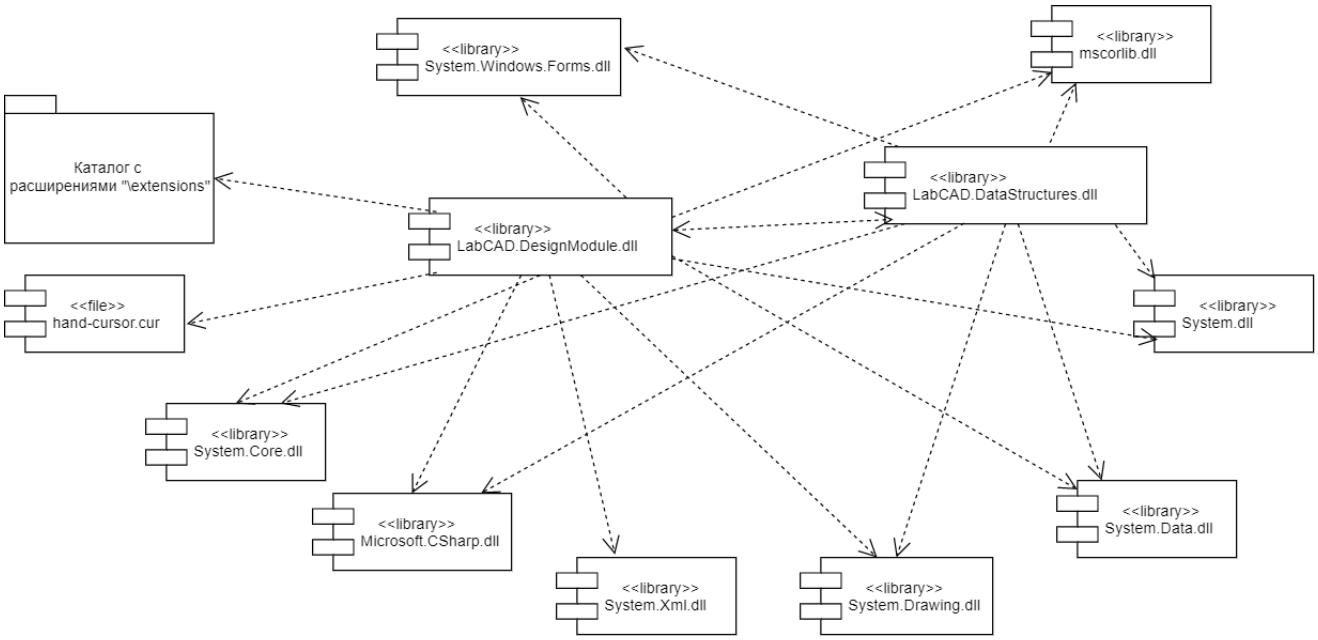


Рисунок К.1 – Контекстная диаграмма компонентов

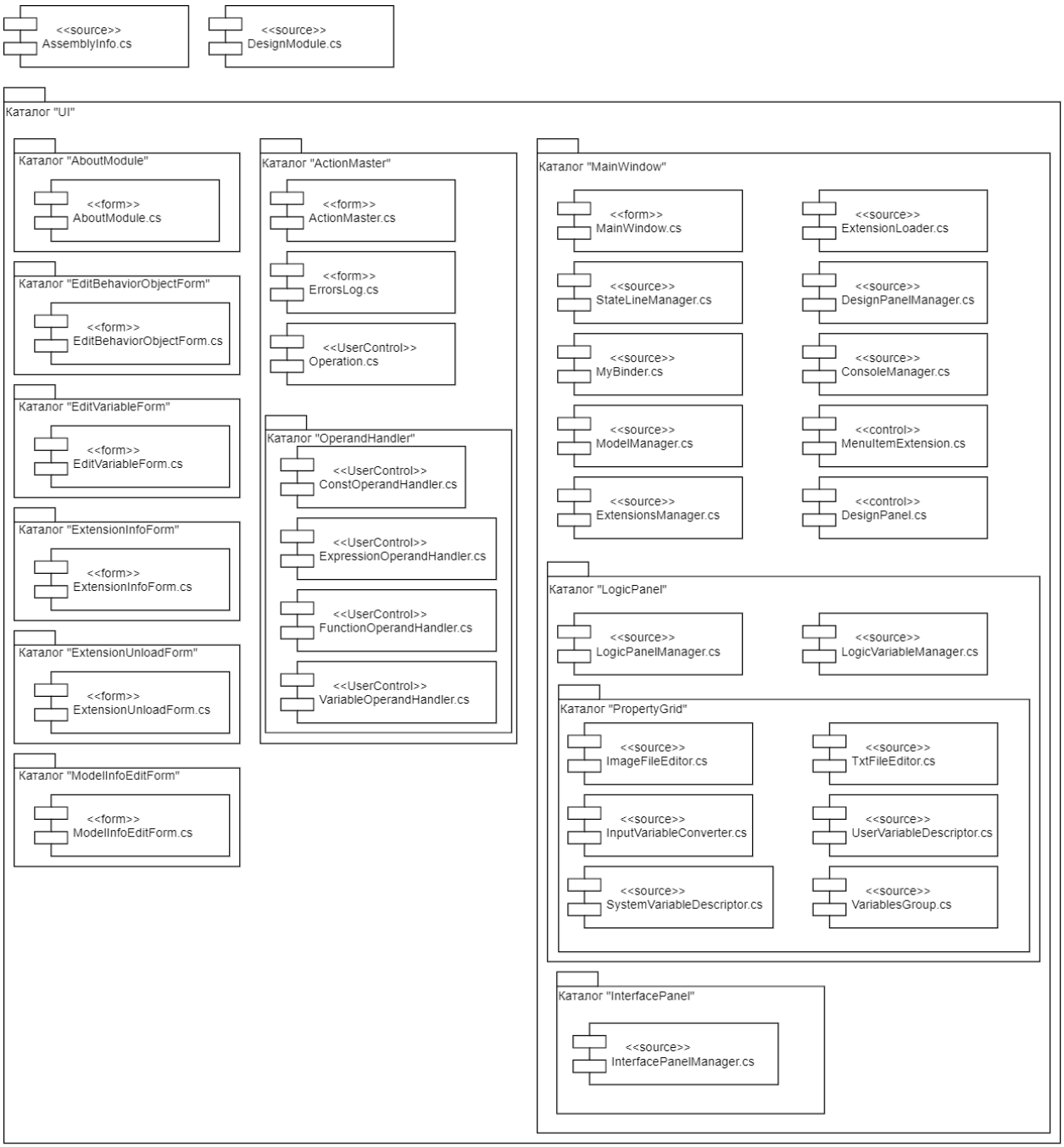


Рисунок К.2 – Диаграмма компонентов библиотеки модуля

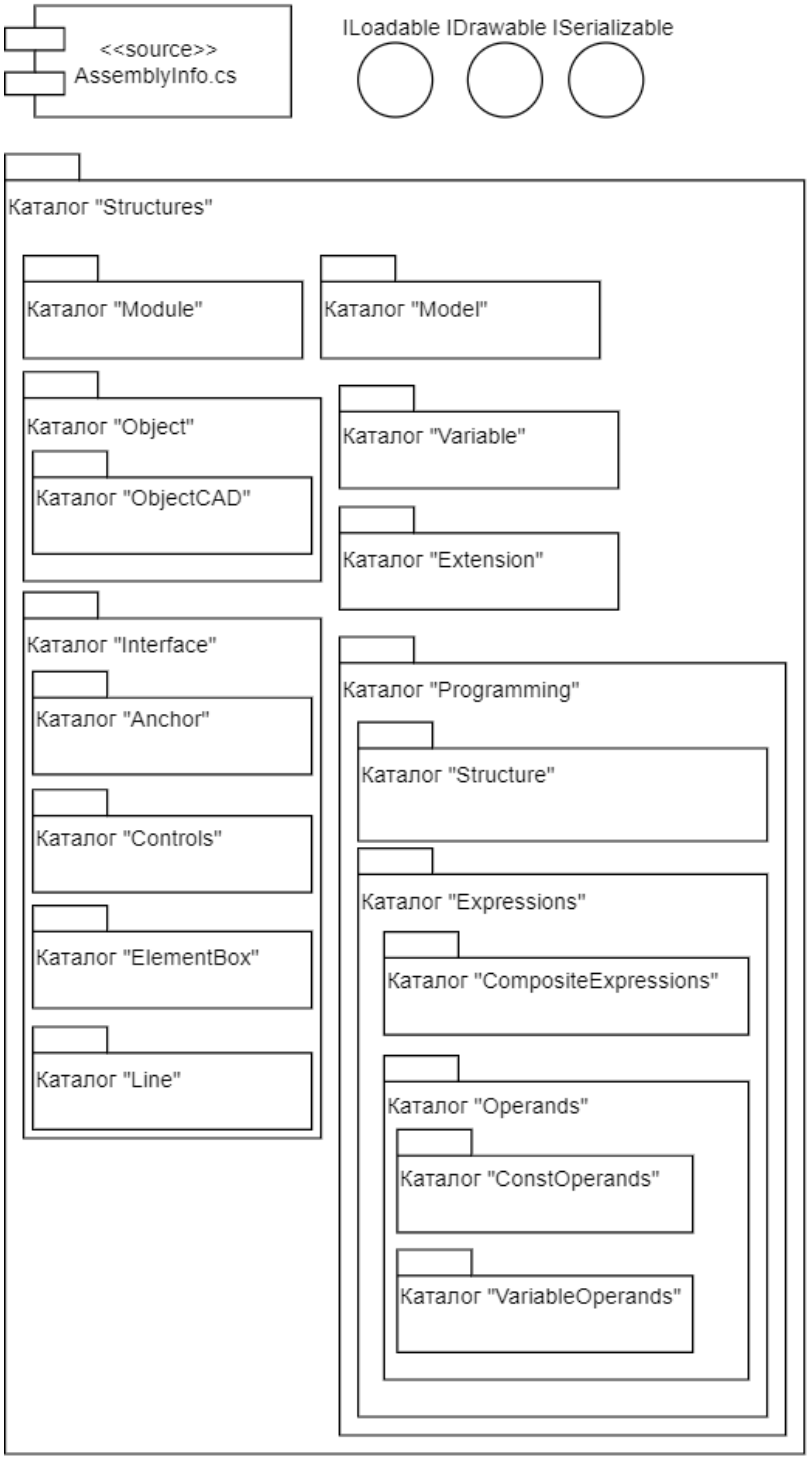


Рисунок К.3 – Диаграмма компонентов библиотеки структур данных

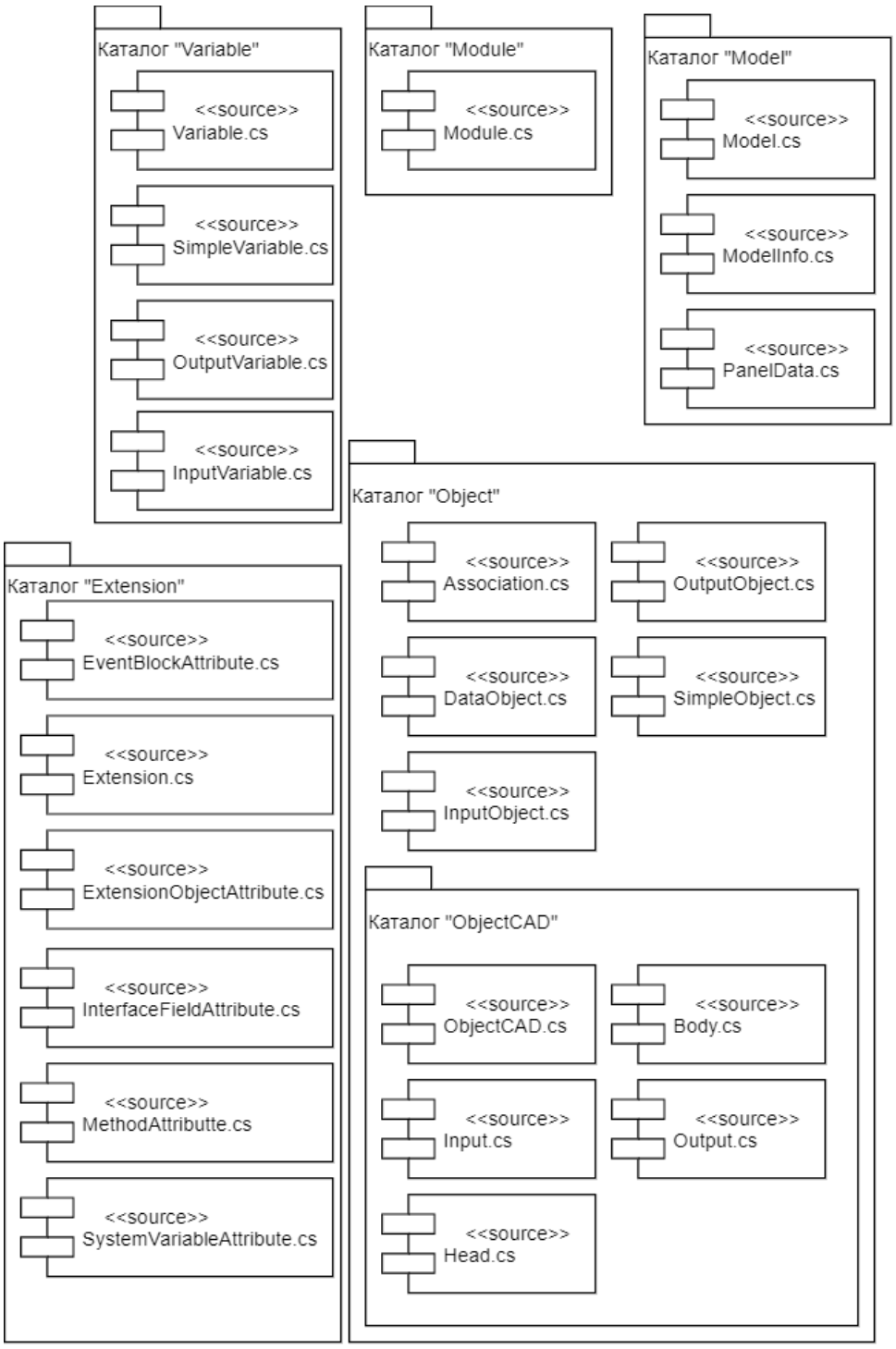


Рисунок К.4 – Диаграмма компонентов каталогов «Variable», «Module», «Model», «Object», «Variable», «Extension»

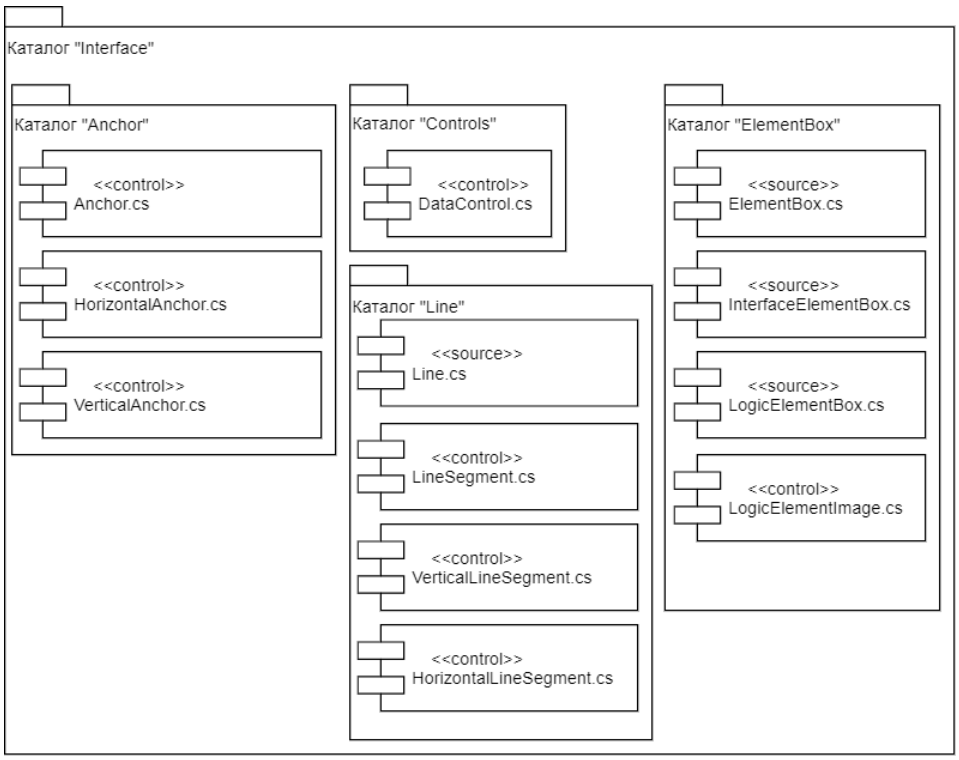


Рисунок К.5 – Диаграмма компонентов каталога «Interface»

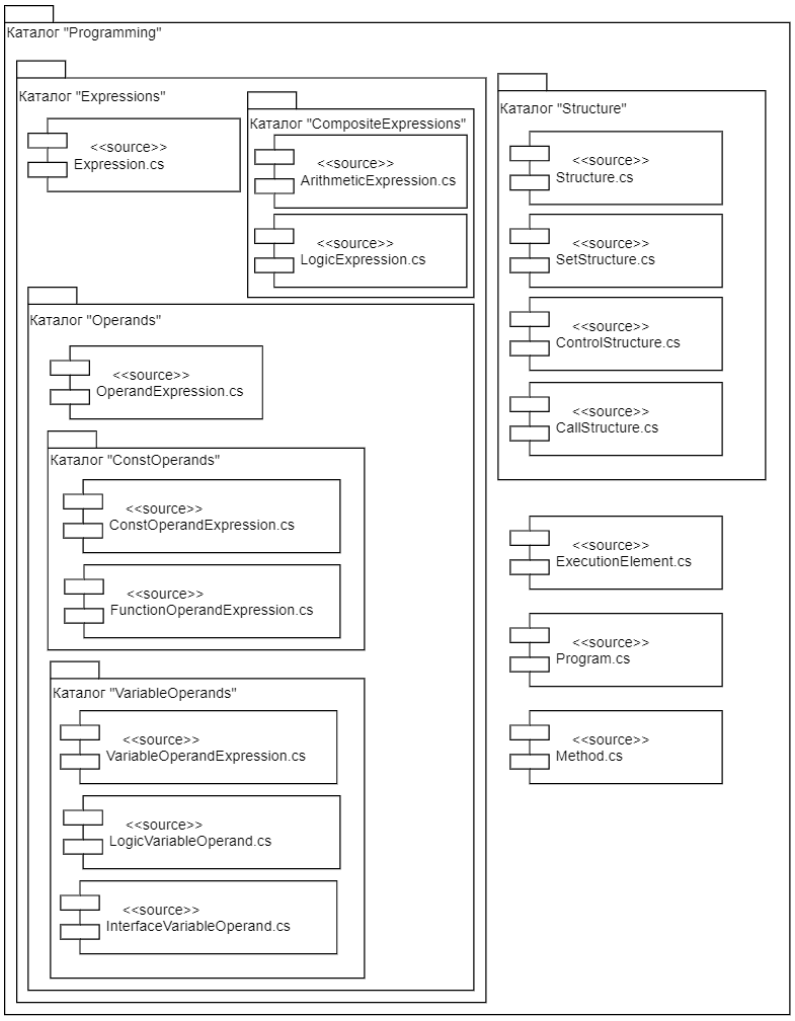


Рисунок К.6 – Диаграмма компонентов каталога «Programming»

Приложение Л
(обязательное)
Фрагменты исходного кода

Листинг М.1 – Исходный код класса менеджера модели

```
using LabCAD.DataStructures.Model;
using LabCAD.DataStructures.Object.ObjectCAD;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms;

namespace LabCAD.Modules.DesignModule
{
    /// <summary>
    /// Класс для работы с моделью
    /// </summary>
    public class ModelManager
    {
        /// <summary>
        /// Состояние сохраненности модели на диске
        /// </summary>
        public bool IsSavedModel { get; private set; } = true;

        /// <summary>
        /// Информация о модели
        /// </summary>
        public ModelInfo CurrentModelInfo { get; private set; }

        /// <summary>
        /// Загрузить модель
        /// </summary>
        public void LoadModel()
        {
            if (IsSavedModel)
            {
                OpenFileDialog dialog = new OpenFileDialog();
                dialog.Filter = "Файлы модели(*.lab)|*.lab";
                if (dialog.ShowDialog() == DialogResult.OK && dialog.FileName != "")
                {
                    LoadFromFile(dialog.FileName);
                }
            }
            else
            {
                DialogResult messageBox = MessageBox.Show("При загрузке модели потеряется
текущая несохраненная модель. Все равно загрузить?", "Внимание",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
                if (messageBox == DialogResult.OK)
                {
                    IsSavedModel = true;
                    LoadModel();
                }
            }
        }
```

```
        /// <summary>
        /// Сохранить модель
        /// </summary>
        public void SaveModel()
        {
            SaveFileDialog dialog = new SaveFileDialog();
            dialog.Filter = "Файлы модели(*.lab)|*.lab";
            if (dialog.ShowDialog() == DialogResult.OK && dialog.FileName != "")
            {
                SaveIntoFile(dialog.FileName);
            }
            else
            {
                MessageBox.Show("Необходимо выбрать файл для сохранения", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }


        /// <summary>
        /// Создать модель
        /// </summary>
        /// <param name="info">Информация о модели</param>
        public void CreateModel(ModelInfo info)
        {
            if (!IsSavedModel)
            {
                DialogResult messageBox = MessageBox.Show("При создании модели потеряется
текущая несохраненная модель. Все равно создать?", "Внимание",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
                if (messageBox != DialogResult.OK)
                {
                    return;
                }
            }
            MainWindow window = ((DesignModule)DesignModule.Current).MainWindow;
            window.LogicPanelManager.Initialize(info.LogicPanelData.PanelSize.Width,
info.LogicPanelData.PanelSize.Height, true);

window.InterfacePanelManager.Initialize(info.InterfacePanelData.PanelSize.Width,
info.InterfacePanelData.PanelSize.Height, true);
            window.LogicPanelManager.Scale = 1.0f;
            window.InterfacePanelManager.Scale = 1.0f;
            CurrentModelInfo = info;
            IsSavedModel = false;
            window.ConsoleManager.AddLine("-> Модель \"" + CurrentModelInfo.Name + "\"
создана.", MessageTypes.Simple);
            window.UpdateModelMenu();
        }

        /// <summary>
        /// Задать инфомацию о модели
        /// </summary>
        /// <param name="info">Информация о модели</param>
        public void SetModelInfo(ModelInfo info)
        {
            MainWindow window = ((DesignModule)DesignModule.Current).MainWindow;
            window.LogicPanelManager.Initialize(info.LogicPanelData.PanelSize.Width,
info.LogicPanelData.PanelSize.Height, false);

window.InterfacePanelManager.Initialize(info.InterfacePanelData.PanelSize.Width,
info.InterfacePanelData.PanelSize.Height, false);
            CurrentModelInfo = info;
            IsSavedModel = false;
```

```
            window.ConsoleManager.AddLine("-> Информация о модели изменена.",
MessageTypes.Simple);
        }

        /// <summary>
        /// Загрузить модель из файла
        /// </summary>
        /// <param name="filePath">Путь к файлу модели</param>
        private void LoadFromFile(string filePath)
        {
            MainWindow window = ((DesignModule)DesignModule.Current).MainWindow;
            try
            {
                window.ConsoleManager.AddLine("-> Загрузка модели...",
MessageTypes.Simple);
                BinaryFormatter binaryFormatter = new BinaryFormatter();
                binaryFormatter.Binder = MyBinder.GetMyBinder();
                Model model;
                Model.Clear();
                using (FileStream fs = new FileStream(filePath, FileMode.Open))
                {
                    model = (Model)binaryFormatter.Deserialize(fs);
                }
                model.Load(window.LogicPanelManager.Panel,
window.InterfacePanelManager.Panel, new object[] { window.LogicPanelManager,
window.InterfacePanelManager });
                IsSavedModel = true;
                CurrentModelInfo = model.ModelInfo;
                window.ConsoleManager.AddLine("-> Модель \"" + CurrentModelInfo.Name +
"\" загружена.", MessageTypes.Simple);
                if(window.IsOpenLogic)

((DesignModule)DesignModule.Current).MainWindow.StateLineManager.SetText("Масштаб: " +
((int)Math.Round(window.LogicPanelManager.Scale * 100)).ToString() + "%",
TypeLineText.Scale);
                else

((DesignModule)DesignModule.Current).MainWindow.StateLineManager.SetText("Масштаб: " +
((int)Math.Round(window.InterfacePanelManager.Scale * 100)).ToString() + "%",
TypeLineText.Scale);
                window.UpdateModelMenu();
            }
            catch (Exception ex)
            {
                window.LogicPanelManager.ClearAll();
                window.InterfacePanelManager.ClearAll();
                IsSavedModel = true;
                CurrentModelInfo = null;
                window.ConsoleManager.AddLine("-> При загрузке модели произошла ошибка: "
+ ex.ToString(), MessageTypes.Error);
            }
        }

        /// <summary>
        /// Сохранить модель в файл
        /// </summary>
        /// <param name="filePath">Путь к файлу модели</param>
        private void SaveIntoFile(string filePath)
        {
            MainWindow window = ((DesignModule)DesignModule.Current).MainWindow;
            try
            {
                window.ConsoleManager.AddLine("-> Сохранение модели...",
MessageTypes.Simple);
                BinaryFormatter binaryFormatter = new BinaryFormatter();
```

```
                using (FileStream fs = new FileStream(filePath, FileMode.Create))
                {
                    binaryFormatter.Serialize(fs, PackModel());
                }
                IsSavedModel = true;
                window.ConsoleManager.AddLine("-> Модель \"" + CurrentModelInfo.Name +
"\" сохранена.", MessageTypes.Simple);
            }
            catch (Exception ex)
            {
                window.ConsoleManager.AddLine("-> При сохранении модели произошла ошибка:
" + ex.ToString(), MessageTypes.Error);
            }
        }

        /// <summary>
        /// Запаковать модель
        /// </summary>
        /// <returns>Объект модели</returns>
        private Model PackModel()
        {
            MainWindow window = ((DesignModule)DesignModule.Current).MainWindow;
            List<object> objects = new List<object>();
            CurrentModelInfo.LogicPanelData.PanelSize = new
Size(window.LogicPanelManager.Width, window.LogicPanelManager.Height);
            CurrentModelInfo.LogicPanelData.PanelScale = window.LogicPanelManager.Scale;
            CurrentModelInfo.InterfacePanelData.PanelSize = new
Size(window.InterfacePanelManager.Width, window.InterfacePanelManager.Height);
            CurrentModelInfo.InterfacePanelData.PanelScale =
window.InterfacePanelManager.Scale;
            foreach (ObjectCAD i in window.LogicPanelManager.Objects)
                objects.Add((object)i);
            return new Model(DesignModule.Current.Version, CurrentModelInfo, objects);
        }
    }
}
```

Листинг М.2 – Исходный код функции генерации конструкции или выражения в Мастере действий

```
        /// <summary>
        /// Получить исполняемый элемент
        /// </summary>
        /// <returns>Исполняемый элемент</returns>
        public ExecutionElement GetExecutionElement()
        {
            ExecutionElement executionElement = null;
            List<Expression> operandExpressions = new List<Expression>();
            List<ArithmeticExpressionType> expressionOperations = new
List<ArithmeticExpressionType>();
            switch (HandlerType)
            {
                case HandlerType.Setting:
                    VariableOperandExpression variableOperandExpression =
((VariableOperandHandler)Elements[0]).GetVariableOperand();
                    Expression value = null;
                    if (OperandsCount == 2)
                    {
                        if (Elements[2].GetType() == typeof(VariableOperandHandler))
                            value =
((VariableOperandHandler)Elements[2]).GetVariableOperand();
                        else if (Elements[2].GetType() ==
typeof(ExpressionOperandHandler))
```

```
                            value =
((ExpressionOperandHandler)Elements[2]).GetExecutionElement();
                        else if (Elements[2].GetType() == typeof(ConstOperandHandler))
                            value =
((ConstOperandHandler)Elements[2]).GetConstOperandExpression();
                        else if (Elements[2].GetType() == typeof(FunctionOperandHandler))
                            value =
((FunctionOperandHandler)Elements[2]).GetFunctionOperandExpression();
                        executionElement = new SetStructure(variableOperandExpression,
value);
                    }

                    else if (OperandsCount > 2)
                    {
                        for (int i = 2; i < Elements.Count; i++)
                        {
                            if (Elements[i].GetType() == typeof(VariableOperandHandler))
                            {

operandExpressions.Add(((VariableOperandHandler)Elements[i]).GetVariableOperand());
                            }
                            else if (Elements[i].GetType() ==
typeof(ExpressionOperandHandler))

operandExpressions.Add(((ExpressionOperandHandler)Elements[i]).GetExecutionElement());
                            else if (Elements[i].GetType() == typeof(Operation))

expressionOperations.Add(GetArithmeticExpressionType(((Operation)Elements[i]).GetValue())
);
                            else if (Elements[i].GetType() ==
typeof(ConstOperandHandler))

operandExpressions.Add(((ConstOperandHandler)Elements[i]).GetConstOperandExpression());
                            else if (Elements[i].GetType() ==
typeof(FunctionOperandHandler))

operandExpressions.Add(((FunctionOperandHandler)Elements[i]).GetFunctionOperandExpression
());
                        }
                        executionElement = new SetStructure(variableOperandExpression,
new ArithmeticExpression(expressionOperations, operandExpressions, FunctionType.None));
                    }
                    break;
                case HandlerType.Arithmetic:
                    foreach (var i in Elements)
                    {
                        if (i.GetType() == typeof(VariableOperandHandler))
                        {

operandExpressions.Add(((VariableOperandHandler)i).GetVariableOperand());
                        }
                        else if (i.GetType() == typeof(ExpressionOperandHandler))

operandExpressions.Add(((ExpressionOperandHandler)i).GetExecutionElement());
                        else if (i.GetType() == typeof(Operation))

expressionOperations.Add(GetArithmeticExpressionType(((Operation)i).GetValue()));
                        else if (i.GetType() == typeof(ConstOperandHandler))

operandExpressions.Add(((ConstOperandHandler)i).GetConstOperandExpression());
                        else if (i.GetType() == typeof(FunctionOperandHandler))

operandExpressions.Add(((FunctionOperandHandler)i).GetFunctionOperandExpression());
                    }
```

```
                    executionElement = new ArithmeticExpression(expressionOperations,
operandExpressions, FunctionType.None);
                    break;
                case HandlerType.Logic:
                    foreach (var i in Elements)
                    {
                        if (i.GetType() == typeof(VariableOperandHandler))
                        {

operandExpressions.Add(((VariableOperandHandler)i).GetVariableOperand());
                        }
                        else if (i.GetType() == typeof(ExpressionOperandHandler))

operandExpressions.Add(((ExpressionOperandHandler)i).GetExecutionElement());
                        else if (i.GetType() == typeof(ConstOperandHandler))

operandExpressions.Add(((ConstOperandHandler)i).GetConstOperandExpression());
                        else if (i.GetType() == typeof(FunctionOperandHandler))

operandExpressions.Add(((FunctionOperandHandler)i).GetFunctionOperandExpression());
                    }
                    executionElement = new
LogicExpression(GetLogicExpressionType(((Operation)Elements[1]).GetValue()),
operandExpressions, FunctionType.None);
                    break;
                case HandlerType.Calling:
                    executionElement = new
CallStructure((FunctionOperandExpression)GetExecutionElement(Elements[0]));
                    break;
                default:
                    executionElement = GetExecutionElement(Elements[0]);
                    break;
            }
            return executionElement;
        }
```

Листинг М.3 – Исходный код класса объекта модели

```
using LabCAD.DataStructures.Interface.ElementBox;
using LabCAD.Modules.DesignModule;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Runtime.Serialization;
using System.Windows.Forms;

namespace LabCAD.DataStructures.Object.ObjectCAD
{

    /// <summary>
    /// Типы базовый объектов
    /// </summary>
    public enum PrimitiveObjectType
    {
        InputObject,
        OutputObject,
        SimpleObject,
        DataObject
    }

    [Serializable]
    /// <summary>
```

```
    /// Представляет собой базовый абстрактный класс объекта, содержащий основные
параметры и методы объекта
    /// </summary>
    public abstract class ObjectCAD : ISerializable, ILoadable, IDrawable
    {
        public static Dictionary<UInt64, ObjectCAD> Objects = new Dictionary<ulong,
ObjectCAD>(); // список объектов для вторичной загрузки
        protected static List<UInt64> _idList = new List<UInt64>(); //список занятых Id

        /// <summary>
        /// Конструктор объекта САПР
        /// </summary>
        /// <param name="primitiveType">Базовый тип объекта</param>
        /// <param name="isBehaviorChange">Возможность изменять алгоритм поведения
объекта</param>
        /// <param name="isVariablesEdit">Возможность редактировать переменные</param>
        /// <param name="extensionId">ID расширения данного объекта САПР</param>
        /// <param name="panel">Панель проектирования логики</param>
        /// <param name="image">Изображение объекта</param>
        /// <param name="pos">Позиция объекта</param>
        /// <param name="size">Размер объекта</param>
        protected ObjectCAD(PrimitiveObjectType primitiveType, bool isBehaviorChange,
bool isVariablesEdit, Guid extensionId, Control panel, Bitmap image, Point pos, Size
size)
        {
            PrimitiveType = primitiveType;
            IsBehaviorChange = isBehaviorChange;
            IsVariablesEdit = isVariablesEdit;
            ExtensionId = extensionId;
            try
            {
                for (UInt64 i = 1; ; i++)
                {
                    if (!_idList.Contains(i))
                    {
                        this.Id = i;
                        _idList.Add(i);
                        break;
                    }
                }
            }
            catch (Exception ex)
            {
                throw new OutOfMemoryException("Вышли за максимальный размер Id");
            }
            Name = "Объект" + Id.ToString();
            LogicBox = new LogicElementBox(this, panel, new LogicElementImage(size, Name,
image), pos, 1.0);
            Image = image;
            Associations = new List<Association>();
            UsedObjects = new List<ulong>();
        }

        protected ObjectCAD(SerializationInfo info, StreamingContext context)
        {
            Id = info.GetUInt64("Id");
            _idList.Add(Id);
            Name = info.GetString("Name");
            PrimitiveType = (PrimitiveObjectType)info.GetInt16("PrimitiveType");
            IsActive = info.GetBoolean("IsActive");
            IsBehaviorChange = info.GetBoolean("IsBehaviorChange");
            IsVariablesEdit = info.GetBoolean("IsVariablesEdit");
            ExtensionId = Guid.Parse(info.GetString("ExtensionId"));
            LogicBox = (LogicElementBox)info.GetValue("LogicBox",
typeof(LogicElementBox));
```

```
            Associations = (List<Association>)info.GetValue("Associations",
typeof(List<Association>));
            Objects.Add(Id, this);
            UsedObjects = (List<UInt64>)info.GetValue("UsedObjects",
typeof(List<UInt64>));
        }

        /// <summary>
        /// Id объекта
        /// </summary>
        public UInt64 Id
        {
            get;
            private set;
        }

        /// <summary>
        /// Имя объекта
        /// </summary>
        public String Name
        {
            get;
            set;
        }

        /// <summary>
        /// Примитивный тип объекта
        /// </summary>
        public PrimitiveObjectType PrimitiveType
        {
            get;
            private set;
        }

        /// <summary>
        /// Возможность изменения алгоритма поведения объекта
        /// </summary>
        public Boolean IsBehaviorChange
        {
            get;
            private set;
        }

        /// <summary>
        /// Состояние активности объекта
        /// </summary>
        public Boolean IsActive
        {
            get;
            set;
        }

        /// <summary>
        /// Возможность изменения переменных объекта
        /// </summary>
        public Boolean IsVariablesEdit
        {
            get;
            private set;
        }

        /// <summary>
        /// Уникальный идентификатор расширения, к которому относится данный объект
        /// </summary>
        public Guid ExtensionId
```

```
        {
            get;
            private set;
        }

        /// <summary>
        /// Заголовок объекта
        /// </summary>
        public Head Head { get; set; }

        /// <summary>
        /// Контейнер объекта логики
        /// </summary>
        public LogicElementBox LogicBox
        {
            get;
            private set;
        }

        /// <summary>
        /// Список ассоциаций
        /// </summary>
        public List<Association> Associations
        {
            get;
            set;
        }

        /// <summary>
        /// Изображение объекта
        /// </summary>
        public virtual Bitmap Image { get; }

        /// <summary>
        /// Существование других объектов в алгоритме поведения текущего объекта
        /// </summary>
        public List<UInt64> UsedObjects { get; set; }

        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)
        {
            info.AddValue("Id", Id);
            info.AddValue("Name", Name);
            info.AddValue("PrimitiveType", (int)PrimitiveType);
            info.AddValue("IsActive", IsActive);
            info.AddValue("IsBehaviorChange", IsBehaviorChange);
            info.AddValue("IsVariablesEdit", IsVariablesEdit);
            info.AddValue("LogicBox", LogicBox);
            info.AddValue("Associations", Associations);
            info.AddValue("ExtensionId", ExtensionId);
            info.AddValue("UsedObjects", UsedObjects);
        }

        public virtual void Load(Panel logicPanel, Panel interfacePanel, object[]
arguments)
        {
            Head.Load(logicPanel, interfacePanel, new object[] { this });
            LogicBox.Load(logicPanel, interfacePanel, new object[] { this });
            foreach (Association i in Associations)
                i.Load(logicPanel, interfacePanel, null);
            ((LogicPanelManager)arguments[0]).AddObject(this);
        }

        /// <summary>
        /// Очистка списка Id
```

```
        /// </summary>
        public static void ClearIds()
        {
            _idList.Clear();
        }


        /// <summary>
        /// Освобождение соответствующего Id
        /// </summary>
        public void RemoveId()
        {
            _idList.Remove(this.Id);
        }
    }
}
```

Приложение М
(обязательное)
Авторская справка

АВТОРСКАЯ СПРАВКА

Я,
Дождиков Игорь Сергеевич
автор выпускной квалификационной работы
Разработка модуля проектирования виртуальных лабораторных стендов

сообщаю, что мне известно о персональной ответственности автора за разглашение сведений, подлежащих защите законами РФ о защите объектов интеллектуальной собственности.

Одновременно сообщаю, что:

1. При подготовке к защите (опубликованию) выпускной квалификационной работы не использованы источники (документы, отчеты, диссертации, литература и т.п.), имеющие гриф секретности или "Для служебного пользования" ВятГУ или другой организации.

2. Данная работа не связана с незавершенными исследованиями или уже с завершенными, но еще официально не разрешенными к опубликованию ВятГУ или другими организациями.

3. Данная работа не содержит коммерческую информацию, способную нанести ущерб интеллектуальной собственности ВятГУ или другой организации.

4. Данная работа не является результатом НИР или ОКР, выполняемой по договору с организацией.

5. В предлагаемом к опубликованию тексте нет данных по незащищенным объектам интеллектуальной собственности других авторов.

6. Согласен на использование результатов своей работы безвозмездно в ВятГУ для учебного процесса, а также на размещение своей работы в электронной информационно-образовательной среде ВятГУ.

7. Использование моей выпускной квалификационной работы в научных исследованиях оформляется в соответствии с законодательством РФ о защите интеллектуальной собственности.

Автор _____ _____
/личная подпись/ /И. О. Фамилия/

"____"_____ 20_____ г.

Сведения по авторской справке подтверждаю:

Заведующий кафедрой _____ _____
/личная подпись/ /И. О. Фамилия/

"____"_____ 20_____ г.

Приложение Н
(обязательное)
Перечень принятых определений и терминов

В настоящей работе используются следующие определения:

- виртуальный лабораторный стенд (ВЛС, стенд) – это многофункциональный программно-методический комплекс, моделирующий действие изучаемых явлений, приборов, механизмов и технических средств;
- проектирование – процесс составления описания, необходимого для создания в заданных условиях еще несуществующего объекта, на основе первичного описания этого объекта и (или) алгоритма его функционирования;
- система автоматизированного проектирования (САПР) – это система, реализующая автоматизированное проектирование;
- модуль САПР – логически связанная совокупность программных компонентов, выполняющая конкретные функции в рамках САПР;
- визуальный язык программирования – совокупность графических или текстографических объектов, манипулируя которыми пользователь задает логику работы модели;
- модель – это совокупность связанных тем или иным способом объектов модели, имеющих список привязанных переменных объекта, ориентированных на взаимодействие с объектами модели и пользователем через задаваемое с помощью визуального языка программирование поведение, предназначенная для моделирования процесса, явления или объекта конкретной предметной области;
- объект модели (объект) – это семантически и программно обособленная единица модели, несущая в себе часть информации в форме переменных объекта и часть логики работы модели в форме поведения объекта;
- ассоциация – связь между объектами модели;
- линия – графическое представление ассоциации на поле проектирования логики;
- контейнер – графическое представление объекта на поле проектирования, с помощью которого пользователь модуля проектирования может взаимодействовать с объектами;
- объект интерфейса – элемент управления графического пользовательского интерфейса, связанный с контейнером интерфейса;
- переменная объекта (переменная) – это программный контейнер, инкапсулирующий в себе значение данной переменной с дополнительной служебной информацией для корректной обработки переменной в контексте модели и для обеспечения взаимодействия с ней пользователя, и содержащий

в себе совокупность методов по выполнению служебных операций с переменной;

- поведение объекта – совокупность программ объекта модели, которые реализуют определенные алгоритмы обработки полученных объектом данных, сохранения результатов обработки и (или) передачи их другим объектам и пользователю;
- программа – это последовательность конструкций нижнего уровня визуального языка программирования LabScript, LabScript for Behaviour;
- цикл работы модели – строго заданная последовательность выполнения программ объектов модели;
- диаграмма сериализации – диаграмма представления структур данных, в которые происходит сериализация объектов классов;
- верхний уровень реализации визуального языка программирования LabScript, LabScript for Objects – подмножество визуального языка программирования LabScript для манипуляции объектами модели: созданием и удалением их, организацией системы связей;
- нижний уровень реализации визуального языка программирования LabScript, LabScript for Behaviour – подмножество визуального языка программирования LabScript для формирования программ для обработки событий в объекте модели;
- конструкция – минимальная независимая синтаксическая единица подмножества LabSript for Behaviour;
- выражение – составная часть конструкций и других выражений, формирующая порядок вычисления значений;
- операнд – составная часть конструкций и выражений, над которыми производятся операции;
- операция – действие, производимое над операндами с целью получения результата вычислений;
- тик – это период времени между двумя последовательными во времени большими циклами работы модели.

Приложение П
(обязательное)
Перечень принятых обозначений и сокращений

В настоящей работе используются следующие сокращения:

- ПЗ – пояснительная записка;
- ДДП – ведомость дипломного проекта;
- ЭВМ – электронно-вычислительная машина;
- ВЛС – виртуальный лабораторный стенд;
- САПР – система автоматизированного проектирования;
- АСУ ТП – автоматизированная система управления технологическим процессом;
- АСНИ – автоматизированная система научных исследований;
- UML – Unified Modeling Language, унифицированный язык моделирования;
- IDEF0 – ICAM DEFinition for Function Modeling, методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов;
- IDEF3 – integrated DEFinition for Process Description Capture Method, методология моделирования и стандарт документирования процессов, происходящих в системе;
- DFD – data flow diagrams, диаграммы потоков данных;
- РБНФ –расширенная Бэкус-Наурова форма;
- ПО – программное обеспечение;
- ВУЗ – высшее учебное заведение;
- ДПЛ – плакат.

Приложение Р
(справочное)
Библиографический список

1. Виртуальные стенды [Электронный ресурс] // Учебная техника и наглядные пособия от производителя, 2020. URL: https://stendlab.ru/virtualnye-laboratornye-stendy (дата обращения: 05.05.2020).

2. Саликаев, Ю. Р. Математические модели и САПР электронных приборов и устройств [Текст]: учебное пособие /Ю. Р. Саликаев – Томск: ТУСУР, 2006. – 182 с.

3. Норенков, И.П. Основы автоматизированного проектирования [Текст]: учеб./ И.П. Норенков – 2 изд., перераб. и доп. – М: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 336 с.: ил.

4. TIOBE Index for May 2020 [Электронный ресурс] // TIOBE – The Software Quaity Company, 2020. URL: https://tiobe.com/tiobe-index/ (дата обращения: 19.05.2020).

5. Псевдо ООП в C [Электронный ресурс] // Хабр, 2020. URL: https://habr.com/ru/post/263547/ (дата обращения: 19.05.2020).

6. Python – это медленно. Почему? [Электронный ресурс] // Блог компании RUVDS.COM, 2020. URL: https://habr.com/ru/company/ruvds/blog/418823/ (дата обращения: 19.05.2020).

7. Руководство по языку C# [Электронный ресурс] // Хранилище документации Майкрософт, 2020. URL: https://docs.microsoft.com/ru-ru/ (дата обращения: 23.05.2020).

8. Как не наступить на грабли, работая с сериализацией [Электронный ресурс] // Блог компании PVS-Studio, 2020. URL: https://habr.com/ru/company/pvs-studio/blog/304734/ (дата обращения: 23.05.2020).