МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Институт информационных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование института (факультета)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МПО ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Разработка и анализ требований\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ наименование дисциплины в соответствии с учебным планом

ЛАБОРАТОРНАЯ РАБОТА № 1, 2, 3

СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Исполнитель

студент \_\_1ПИб-02-1оп-22\_\_

группа

\_\_\_Микуцких Г. А.\_\_\_

Фамилия, имя, отчество

Руководитель \_\_\_Ершов Е.В.\_\_\_

Ф.И.О. преподавателя

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_2025\_ год

## Лабораторная работа №1

* 1. IDEF0

Методология функционального моделирования – это технология описания системы в целом как множества взаимозависимых действий, или функций. Функции системы исследуются независимо от объектов, которые обеспечивают их выполнение.

Одна из наиболее важных особенностей нотации IDEF0 – постепенное введение все более детальных представлений модели системы по мере разработки отдельных диаграмм. Построение модели IDEF0 начинается с представления всей системы в виде простейшей диаграммы, состоящей из одного блока процесса и стрелок ICOM, служащих для изображения основных видов взаимодействия с объектами вне системы. Поскольку исходный процесс представляет всю систему как единое целое, данное представление является наиболее общим и подлежит дальнейшей декомпозиции. Такая диаграмма называется контекстной. Функции изображаются на диаграммах как поименованные прямоугольники. Имя функции должно быть выражено отглагольным существительным, обозначающим действие, например, «Изготовление детали».

Во-первых, определяется назначение модели – набор вопросов, на которые должна отвечать модель, а также границы моделирования – ширины охвата предметной области и глубины детализации.

Во-вторых, для построения модели необходимо чётко сформулировать цель (Purpose), которая должна отвечать на следующие вопросы:

1. Почему этот процесс должен быть смоделирован?
2. Что должна показывать модель?
3. Что может получить читатель?

В-третьих, выбирается точка зрения (ViewPort), соответствующая цели моделирования. Однажды выбранная точка зрения остаётся неизменной для всех элементов модели.

Все функциональные блоки IDEF0 нумеруются. Номер состоит из префикса произвольной длины и номера блока. Контекстный блок всегда имеет номер 0. Для каждого уровня декомпозиции в конец номера добавляется одна цифра. Если функция не декомпозирована, то в левом верхнем углу изображается диагональная черта.

Взаимодействие функций описывается в виде стрелок. В IDEF0 различают пять типов стрелок:

1. вход (Input) – материал или информация, которые используются или преобразуются функцией для получения результата (выхода). Стрелки входа всегда направлены в левую сторону функционального блока;
2. управление (Control) – правила, инструкции, стандарты, влияющие на работу функционального блока, но не преобразующиеся им. Стрелка управления изображается входящей в верхнюю грань функционального блока;
3. выход (Output) – материал или информация, полученные в результате работы функционального блока. Изображается исходящей из правой грани функционального блока;
4. механизм (Mechanism) – ресурсы, выполняющие моделируемое действие. Стрелка изображается входящей в нижнюю грань функционального блока (связь выход-механизм);
5. вызов (Call) – специальная стрелка, указывающая на другую модель. Исходит из нижней грани функционального блока.

В конечном итоге модель IDEF0 представляет собой набор иерархически взаимосвязанных диаграмм с сопроводительной документацией, которая разбивает исходное представление сложной системы на отдельные составные части. Детали каждого основного процесса представляются в виде более подробных процессов на других диаграммах.

То с какой стороны дуга подходит к блоку является своего рода значением данной дуги: слева – вход в блок, справа – выход в блок, сверху – управляющая информация, снизу – механизмы (средства производства).

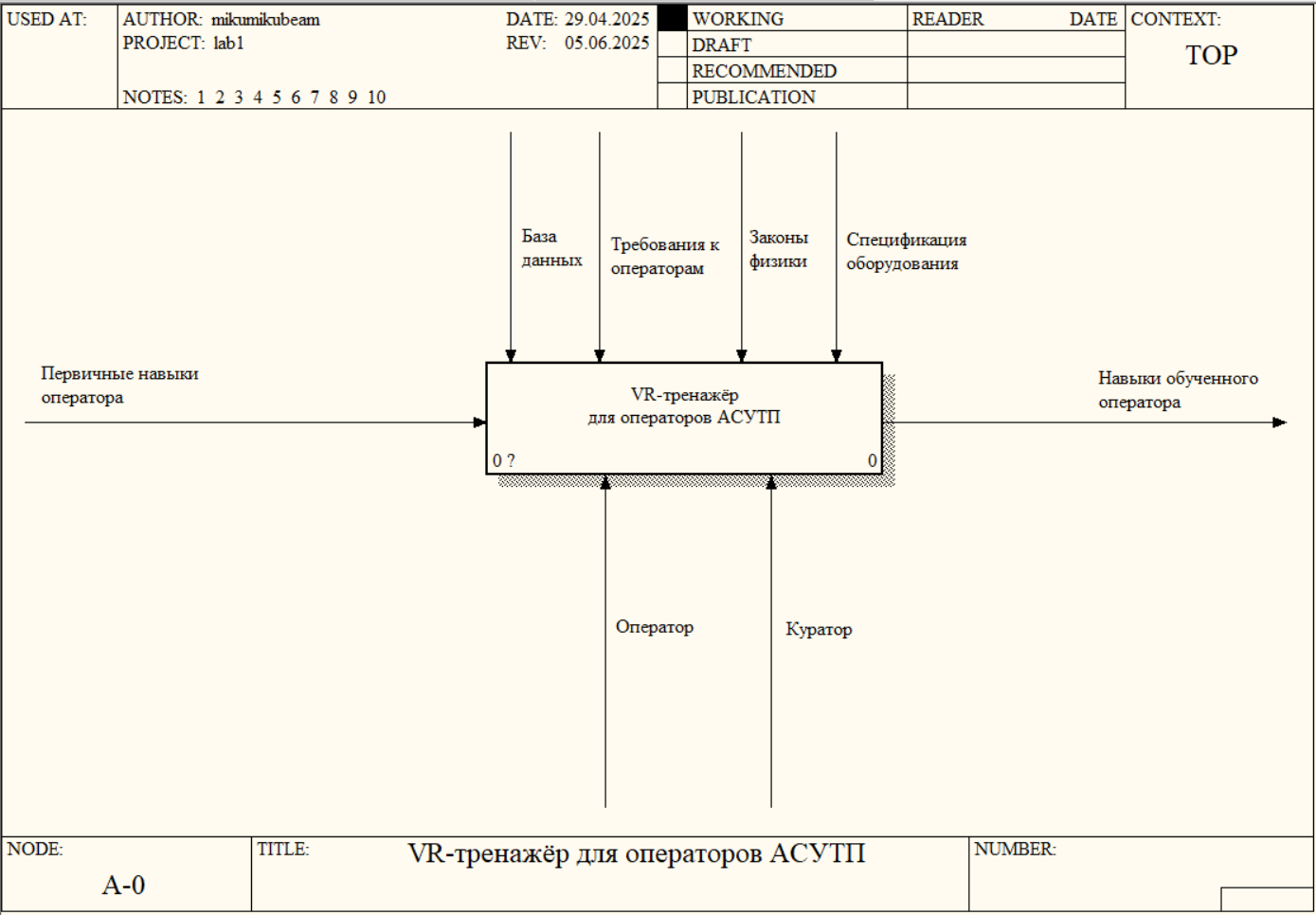


Рис. 1. Контекстная диаграмма IDEF0

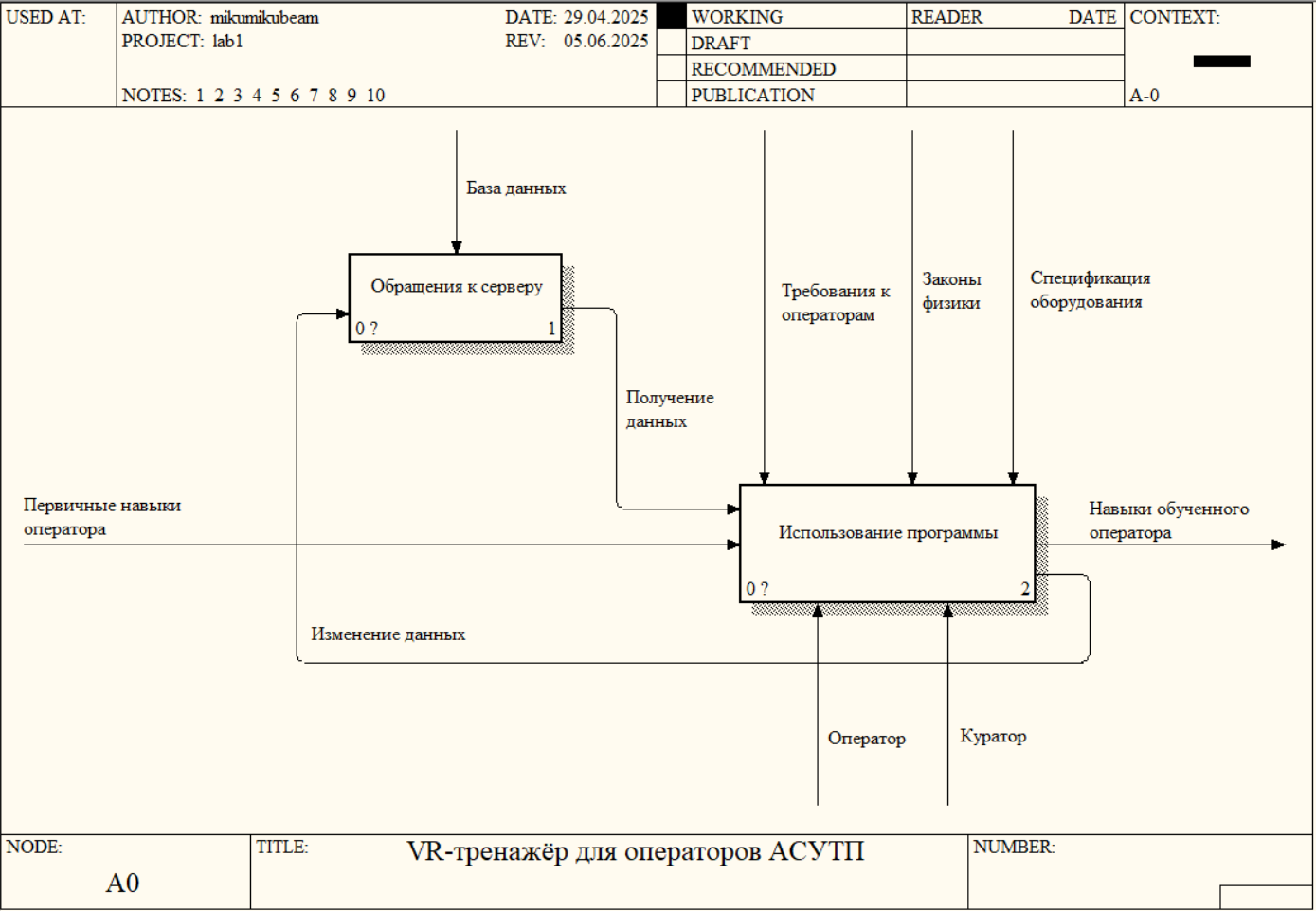


Рис. 2. Декомпозированный функциональный блок «VR-тренажёр»

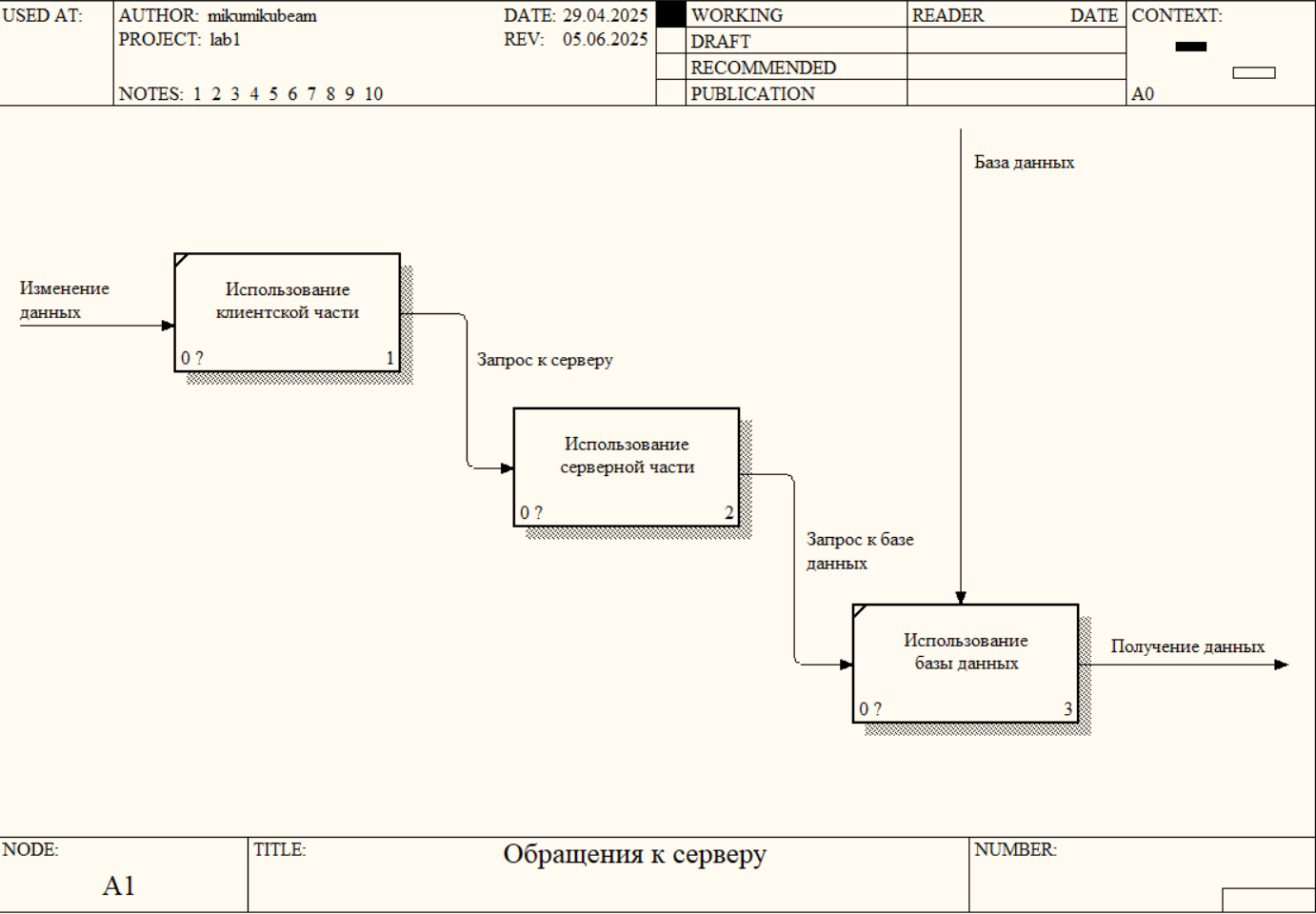


Рис. 3. Декомпозиция функционального блока «Обращения к серверу»

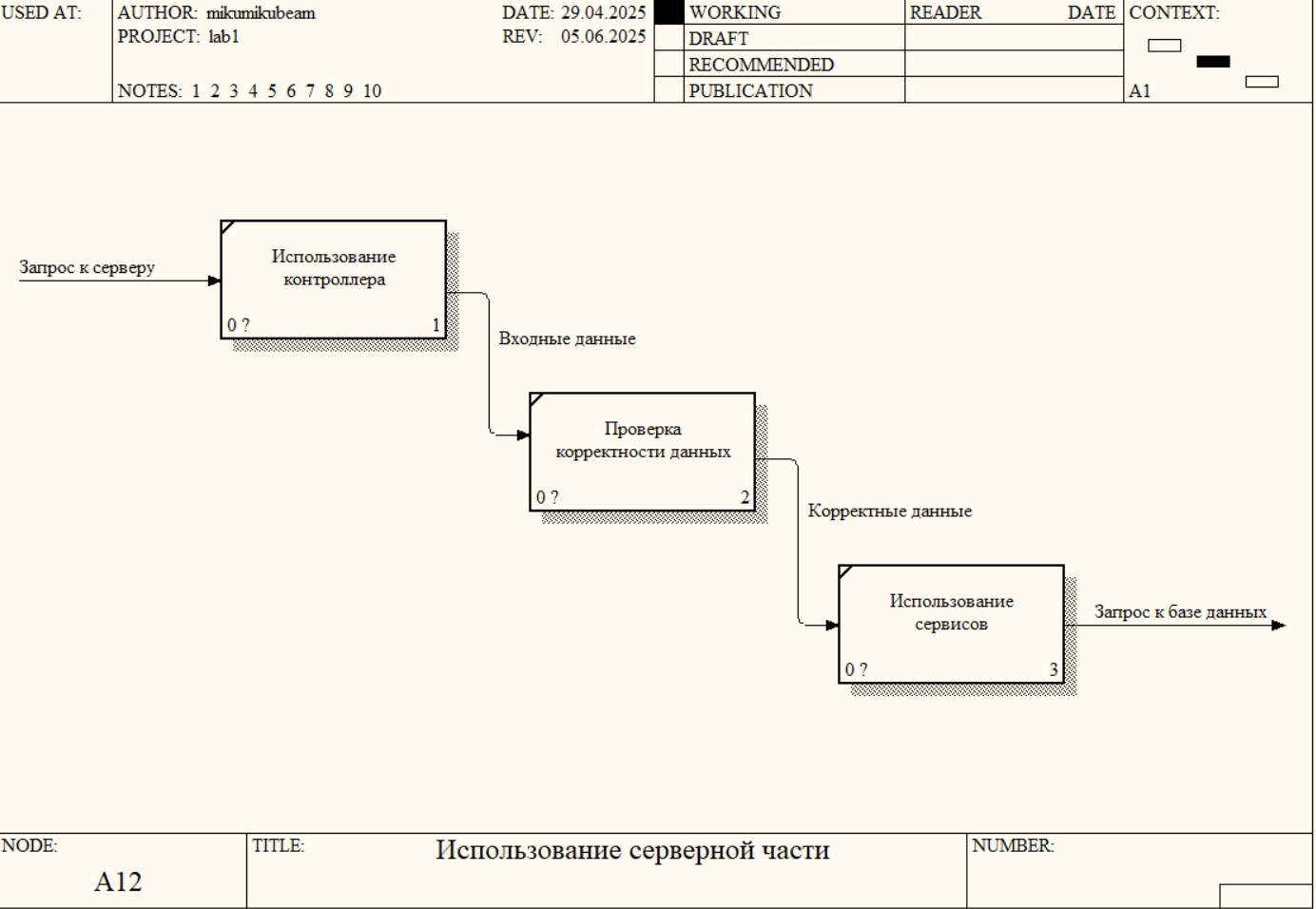


Рис. 4. Декомпозиция блока «Использование серверной части»

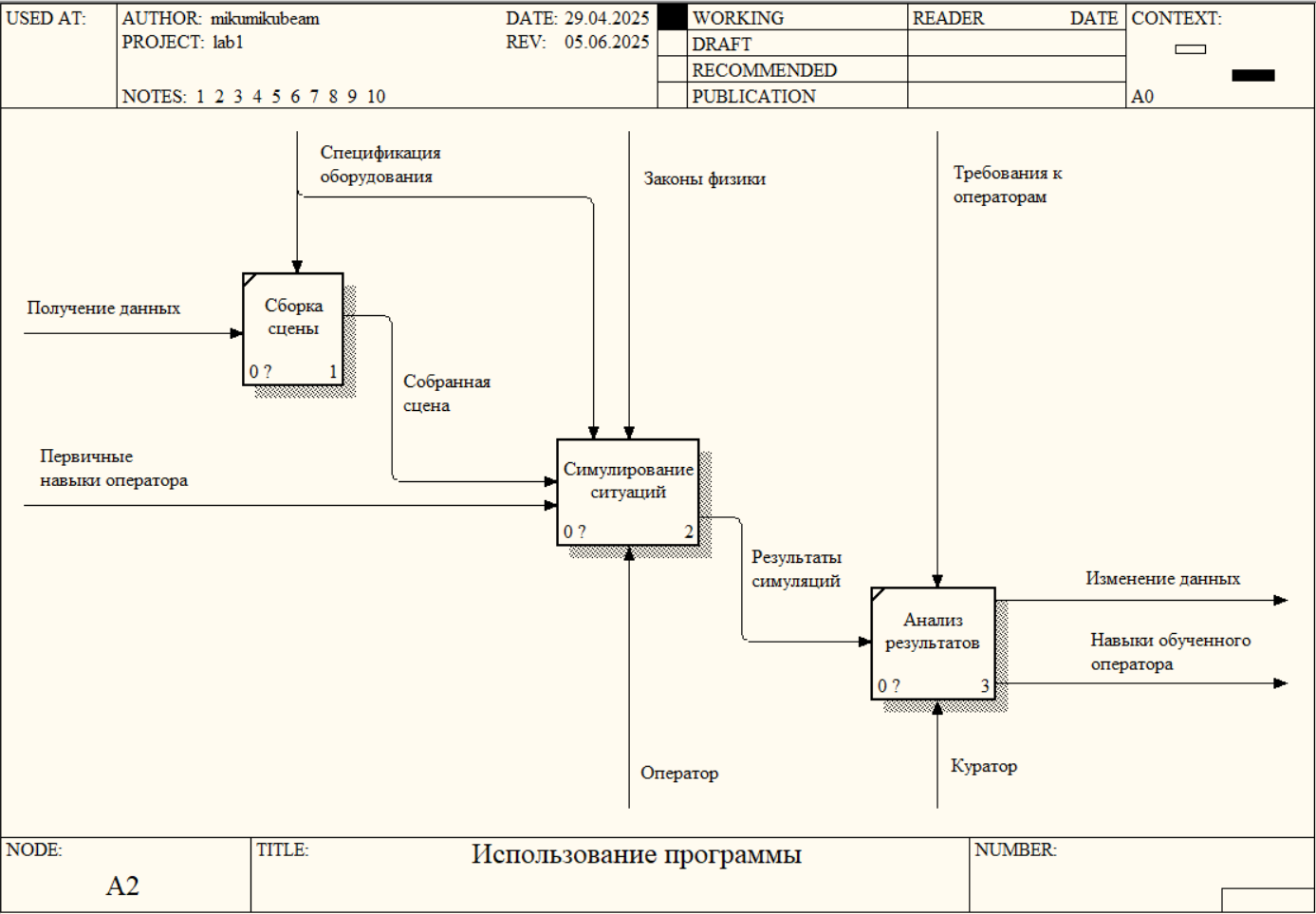


Рис. 5. Декомпозиция функционального блока «Использование программы»

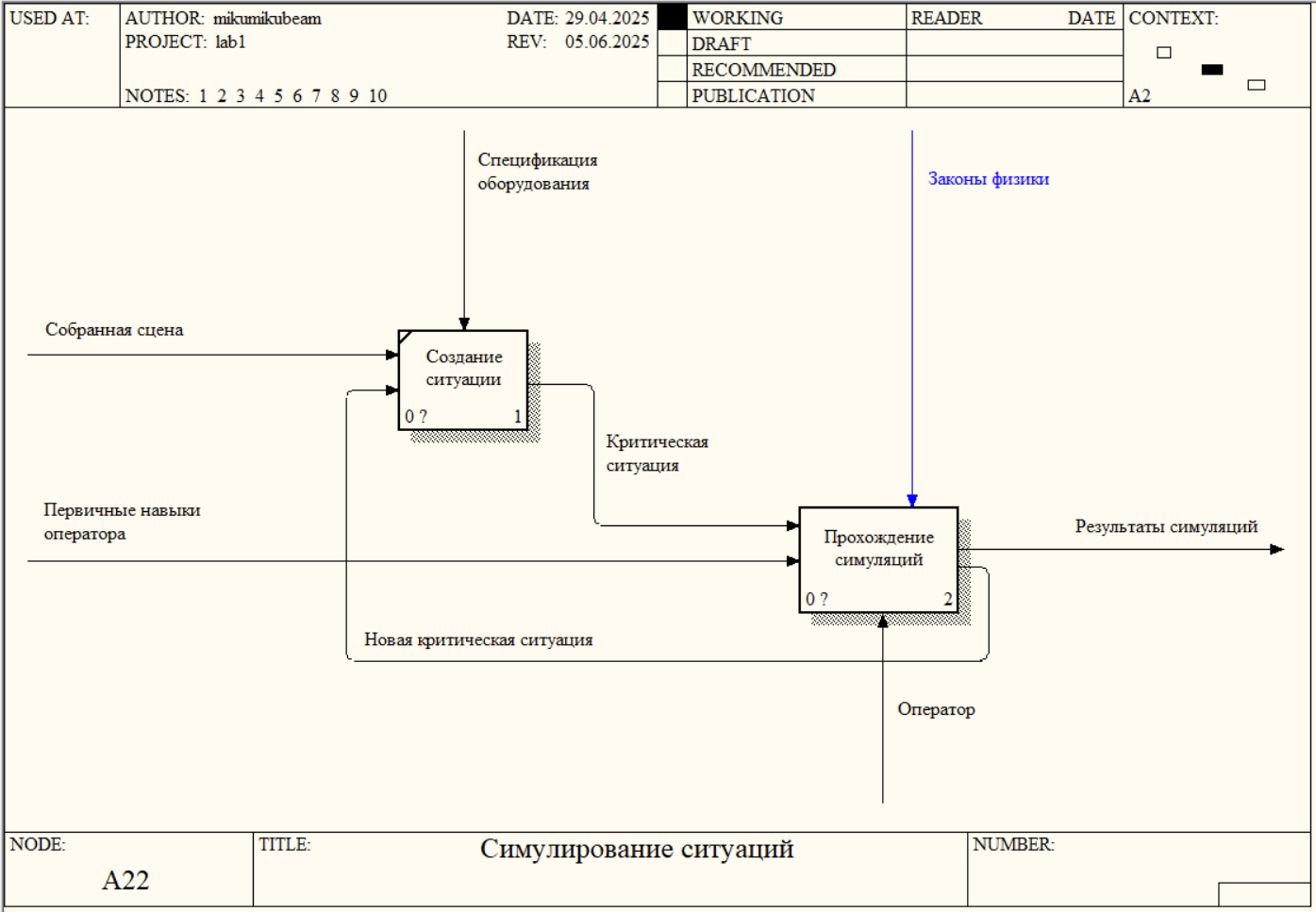


Рис. 6. Декомпозированный функциональный блок «Симулирование ситуаций»

Функция «Прохождение симуляций» декомпозирована в IDEF3.

Таблица 1

Описание элементарных функций IDEF0

|  |  |
| --- | --- |
| **Элементарная функция** | **Описание** |
| Использование клиентской части | Набор способов ввода данных в базу данных и отображения получаемой информации |
| Использование базы данных | Комплекс методов для обращения непосредственно к базе данных |
| Использование контроллеров | Комплекс методов и обращений для решения задач запроса |
| Проверка корректности данных | Валидация входных данных |
| Использование сервисов | Перевод запроса на методы взаимодействия с базой данных |
| Создание моделей оборудования | Процесс создания виртуальных моделей, имитирующих характеристики и поведение реального оборудования |
| Сборка сцены | Процесс создания рабочей среды симуляции |
| Анализ результатов | Анализ результатов пройдённых симуляций, который записывается в БД и отправляется куратору |
| Создание ситуации | Симулирование отклонений в работе оборудования |

Таблица 2

Словарь терминов IDEF0

|  |  |
| --- | --- |
| **Термин** | **Описание** |
| База данных | Набор программ, осуществляющих доступ к базе данных |
| Требования к операторам | Методические указания и правила охраны труда |
| Законы физики | Законы физики, в большей степени оказывающие влияние на изменение состояния оборудования |
| Спецификация оборудования | Законы поведения оборудования |
| Первичные навыки оператора | Навыки сотрудника, проходящего обучение |
| Навыки обученного оператора | Компетенции, полученные после прохождения симуляции |
| Запрос к серверу | Набор данных для работы с сервером |
| Запрос к базе данных | Набор обработанных данных из сервера |
| Входные данные | Данные из запроса |
| Корректные данные | Провалидированные входные данные |
| Получение данных | Загрузка необходимых данных из базы данных |
| Собранная сцена | Рабочая среда с моделями оборудования |
| Результаты симуляций | Результат прохождения симуляции |
| Изменение данных | Загрузка результатов в базу данных |
| Критическая ситуация | Комплекс отклонений от нормы, который может поставить работу предприятия под угрозу |
| Новая критическая ситуация | Изменение сложности симуляции на основе входных данных об успехе похождения предыдущих этапов |

* 1. IDEF3

IDEF3 – способ описания процессов, дополняющий IDEF0 и содержащий все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа. В отличие от большинства технологий моделирования бизнес-процессов, IDEF3 не имеет жестких синтаксических или семантических ограничений.

Основой модели IDEF3 служит так называемый сценарий бизнес-процесса, который определяет назначение и границы модели. Для названия сценария используют глаголы и отглагольные существительные.

Как и в любой технологии моделирования действий, главной организационной единицей модели IDEF3 является диаграмма. Центральный компонент модели – подпроцесс или действие, отображаемый на диаграмме в виде прямоугольника. Каждому действию присваивается номер, которому предшествует номер его родителя.

Существенные взаимоотношения между процессами выделяются связями. В IDEF3 все связи являются однонаправленными.

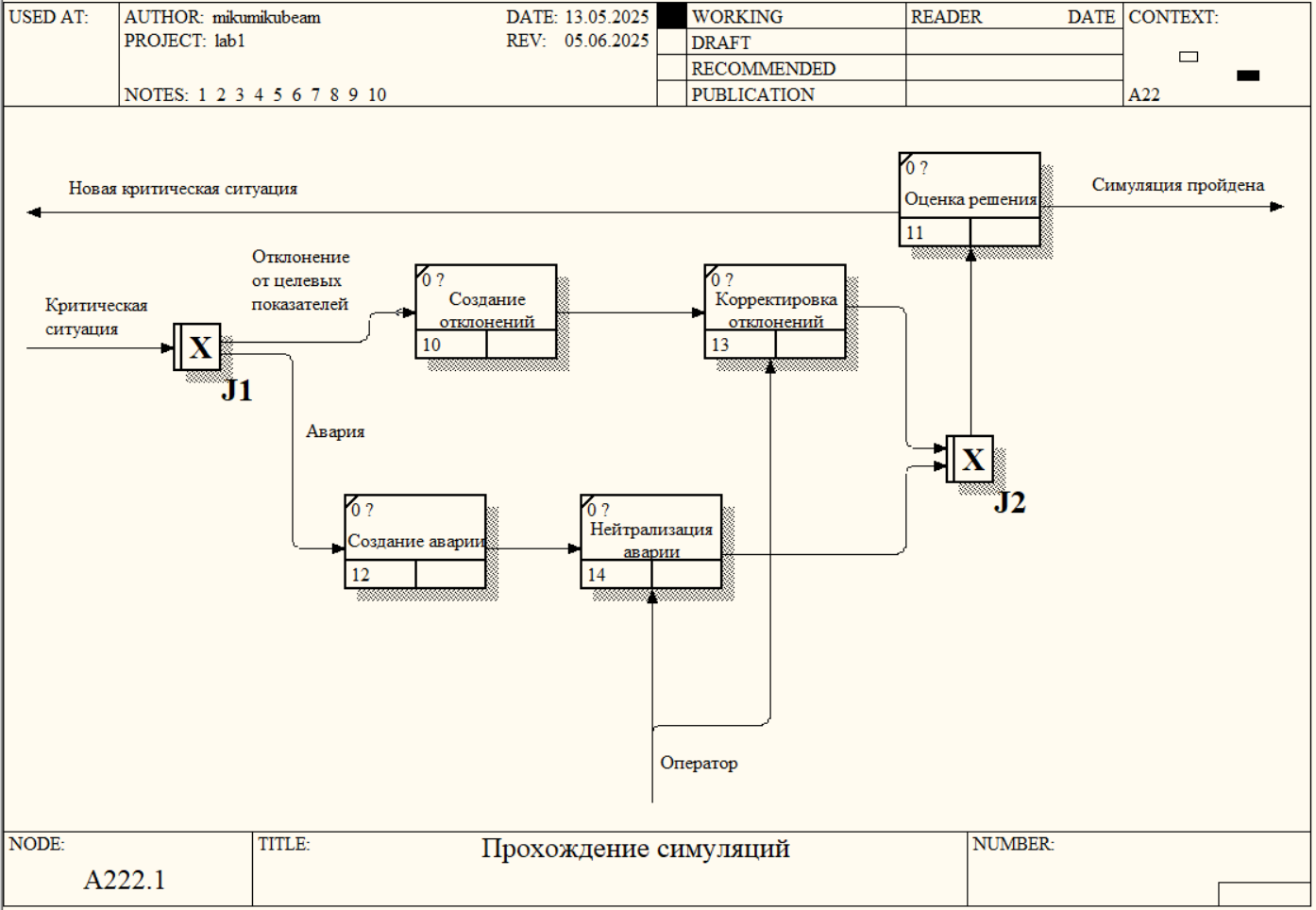


Рис. 7. Декомпозированный блок «Прохождения симуляции»

Время от времени будут возникать отклонения показателей на моделях оборудования. Если показатели не будут в пределах нормы на протяжении длительного времени или оператором будут предприняты неверные решения, то создаётся новая критическая ситуация, в частности авария.

* 1. DFD

В основе графического моделирования информационных систем с помощью диаграмм потоков данных лежит специальная технология построения диаграмм потоков данных DFD.

В DFD процессы представляют собой функции системы, преобразующие входы и выходы. Хотя функции изображаются прямоугольниками с закруглёнными углами, смысл их совпадает со смыслом функций IDEF0 и IDEF3. Внешние сущности изображаются в виде прямоугольников и обозначают входы в систему и/или выходы из системы. Стрелки в DFD показывают, как объекты реально перемещаются от одного действия к другому. Стрелки могут разветвляться и соединяться, что позволяет описать декомпозицию стрелок. Хранилища данных изображают объекты в состоянии покоя.

Представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы – движение объектов, хранение, поставка и распространение.

Диаграммы DFD могут быть построены с использованием традиционного структурного анализа (подобно построению IDEF0). Альтернативный подход построения DFD-моделей называется событийным разделением, в котором различные диаграммы DFD выстраивают модель системы.

|  |  |
| --- | --- |
| Работы | Представляют собой функции системы, преобразующие входы в выходы. Изображаются прямоугольниками со скруглёнными углами. |
| Внешние сущности | Изображают входы в систему и/или выходы из системы. Изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. |
| Стрелки (Потоки данных) | Описывают движение объектов из одной части системы в другую. Стрелки могут подходить и выходить из любой грани прямоугольника работы. Также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью и между внешними сущностями. |
| Хранилище данных | Изображают объекты в покое, в отличие от стрелок, описывающих объекты в движении. Это очереди и т. п. |

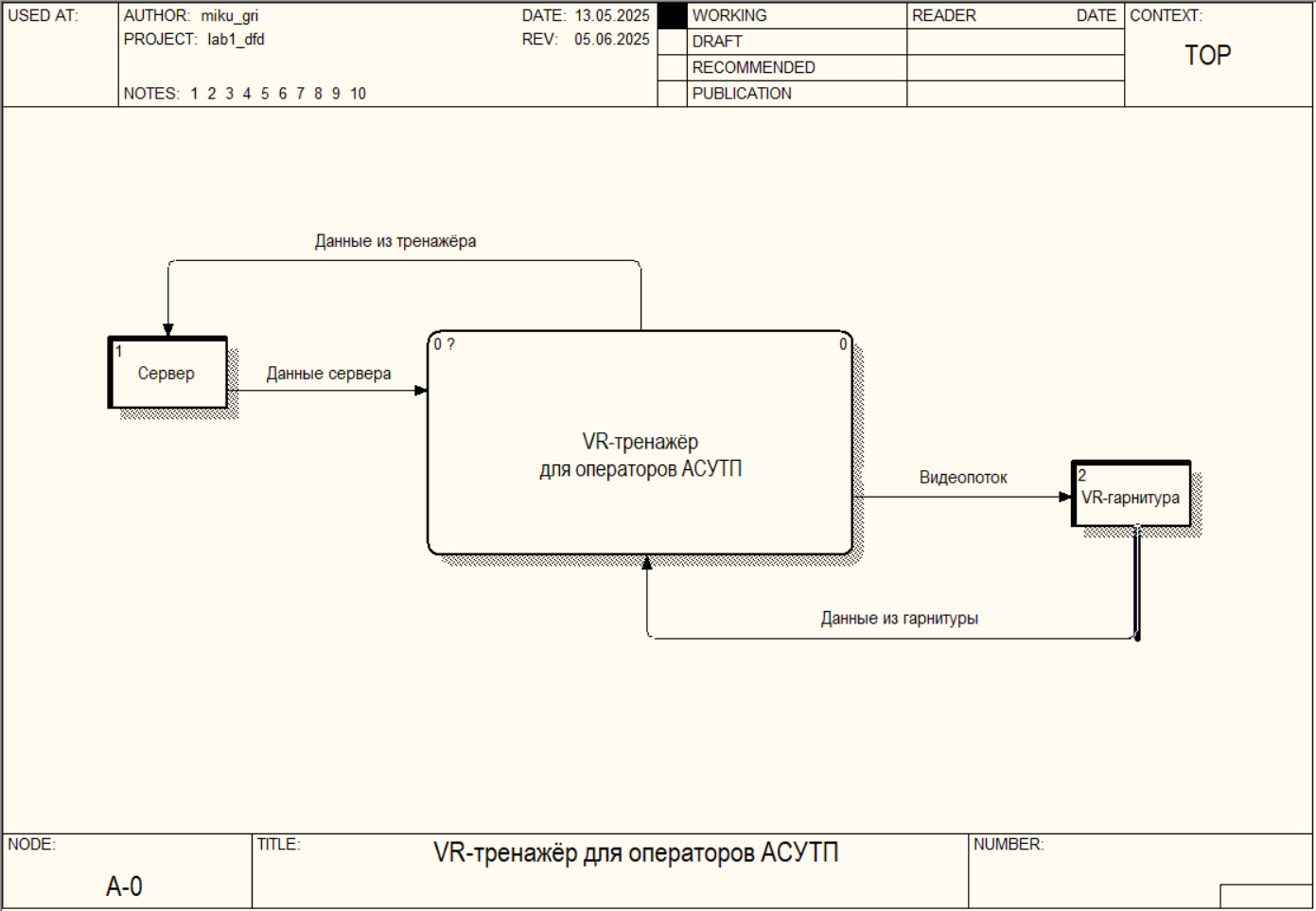


Рис. 8. DFD-диаграмма «VR-тренажёр для операторов АСУТП»

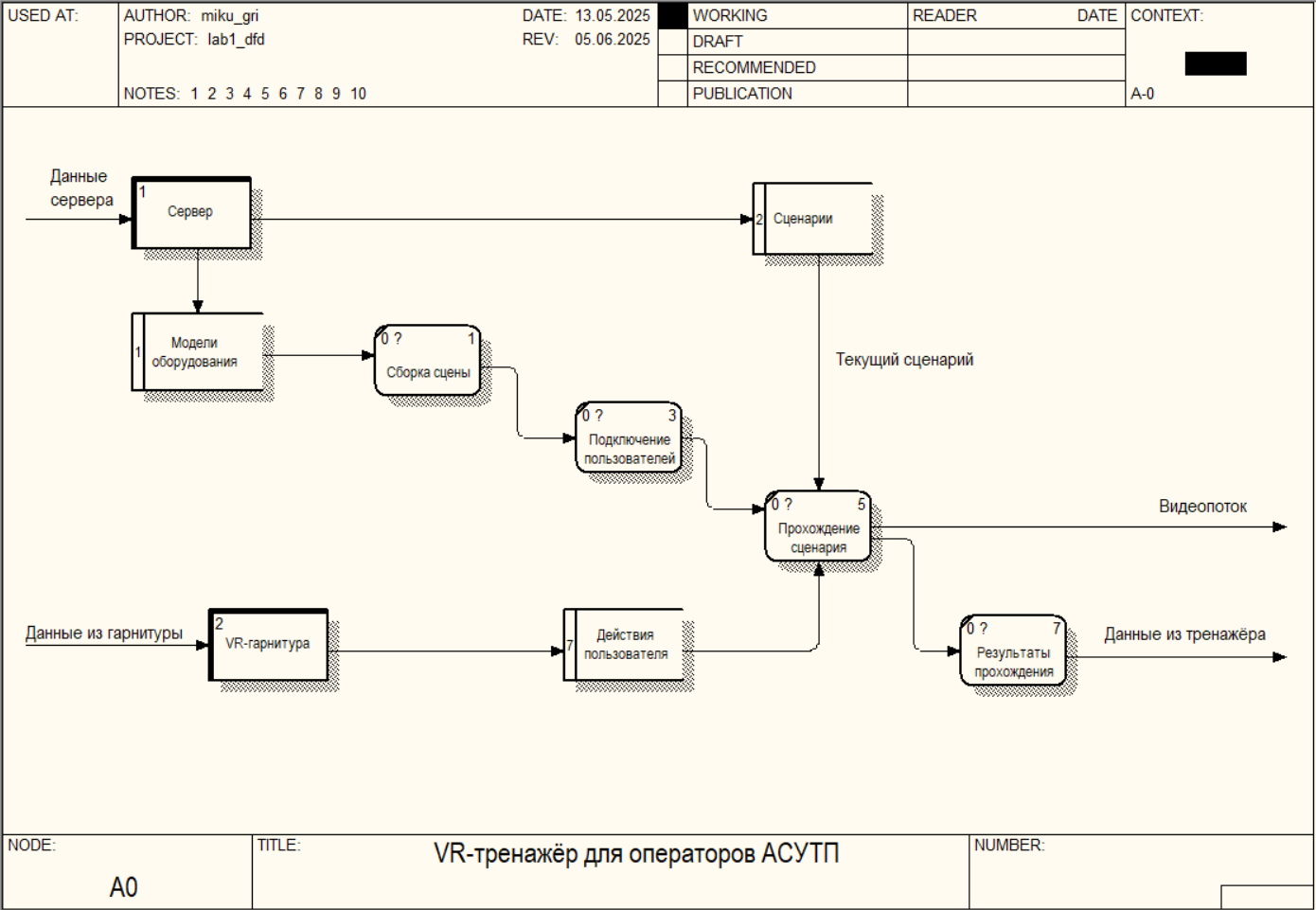


Рис. 9. Декомпозированная DFD-диаграммы

Таблица 3

Описание элементарных процессов DFD

|  |  |
| --- | --- |
| **Элементарный процесс** | **Описание** |
| Сборка сцены | Загружается сцена с определенным сценарием её прохождения (этап работы имитация рабочего места) |
| Подключение пользователей | Появление пользователей на сцене и их синхронизация через сервер |
| Прохождение сценария | Прохождение пользователями всех этапов сценариев. По необходимости, повтор или генерация нового |
| Результаты прохождения | Вывод детальных результатов прохождения |

Таблица 4

Описание внешних сущностей DFD

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Сервер | Выделенный сервер для реализации многопользовательского режима |
| VR-гарнитура | Оборудование для отображения сцены, положения других пользователей и их действий на сцене, результатов прохождения сцены |

Таблица 5

Словарь терминов DFD

|  |  |
| --- | --- |
| **Термин** | **Описание** |
| Модели оборудования | Набор виртуальных моделей оборудования |
| Сценарии | Набор поведений оборудования |
| Действия пользователей | Влияние, оказываемое пользователем на окружение, положение VR-гарнитуры в пространстве |
| Видеопоток | Изображение, передаваемое на устройство. Баллы, полученные в ходе прохождения сценария, показатели оборудования и прочее |
| Данные из тренажёра | Передаются данные о состоянии объектов симуляции:  положение объектов, положение пользователей, показатели моделей |



Рис. 10. Общая структура моделей

## Лабораторная работа №2

## Разработка структурной и функциональной схемы.

Процесс проектирования сложного программного обеспечения начинают с уточнения его структуры, то есть определения структурных компонентов и связей между ними. Структурной называют схему, отражающую состав и взаимодействие по управлению частей разрабатываемого ПО.

Структурная схема программной системы, как правило, показывает наличие подсистем или других структурных компонентов. В отличие от программного комплекса отдельные части (подсистемы) программной системы интенсивно обмениваются данными между собой и, возможно, с основной программой. Структурная же схема программной системы этого обычно не показывает.

Структурный подход к программированию предлагал ранее осуществлять декомпозицию программ методом пошаговой детализации. Результатом декомпозиции является структурная схема программы, которая представляет собой многоуровневую иерархическую схему взаимодействия подпрограмм по управлению.

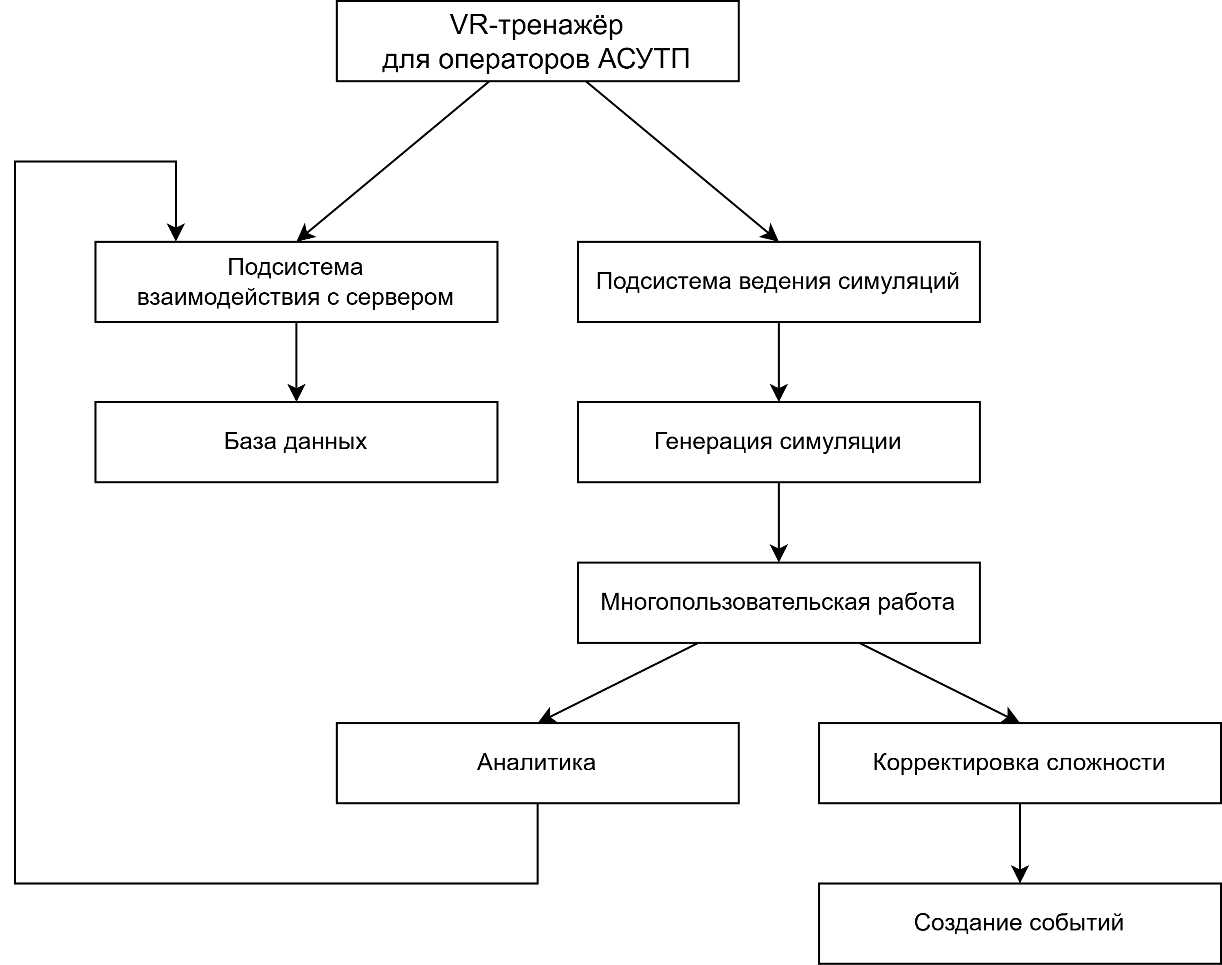


Рис. 11. Структурная схема VR-тренажёра

Таблица 6

Описание структурной схемы

|  |  |
| --- | --- |
| **Название блока** | **Описание** |
| Подсистема взаимодействия с сервером | Управляет сохранением, извлечением и удалением данных в базе данных |
| База данных | Хранилище данных о результатах симуляций и пользователей |
| Подсистема ведения симуляций | Комплекс ведения, управления и аналитики симуляций операторов АСУТП |
| Аналитика | Обрабатывает данные из симуляции и базы данных, выносит закономерности и рекомендации |
| Многопользовательская работа | Обрабатывает многопользовательские сессии и синхронизирует действия между операторами |
| Генерация симуляции | Создаёт, модифицирует и ведёт симуляции согласно определённым сценариям |
| Корректировка сложности | Подстраивает сложность симуляции под успешность прохождения |
| Сборка сцены | Собирает и настраивает параметры сцены согласно сценарию |
| Создание событий | Создаёт отклонения и критические ситуации согласно сценарию |

Функциональная схема или схема данных (ГОСТ 19.701-90) – схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств. Для изображения функциональных схем используют специальные обозначения, установленные стандартом.

При структурном подходе особенно тщательно необходимо прорабатывать спецификации межпрограммных интерфейсов, так как от качества их описания зависит количество самых дорогостоящих ошибок. К самым дорогим относятся ошибки, обнаруживаемые при комплексном тестировании, так как для их устранения могут потребоваться серьёзные изменения уже отлаженных текстов. Функциональными называют диаграммы, в первую очередь отражающие взаимосвязи функций разрабатываемого программного обеспечения

Отображение взаимосвязи функций активностной модели осуществляется посредством построения иерархии функциональных диаграмм, схематически представляющих взаимосвязи нескольких функций. Каждый блок такой диаграммы соответствует некоторой функции, для которой должны быть определены: исходные данные, результаты, управляющая информация и механизмы её осуществления — человек или технические средства.

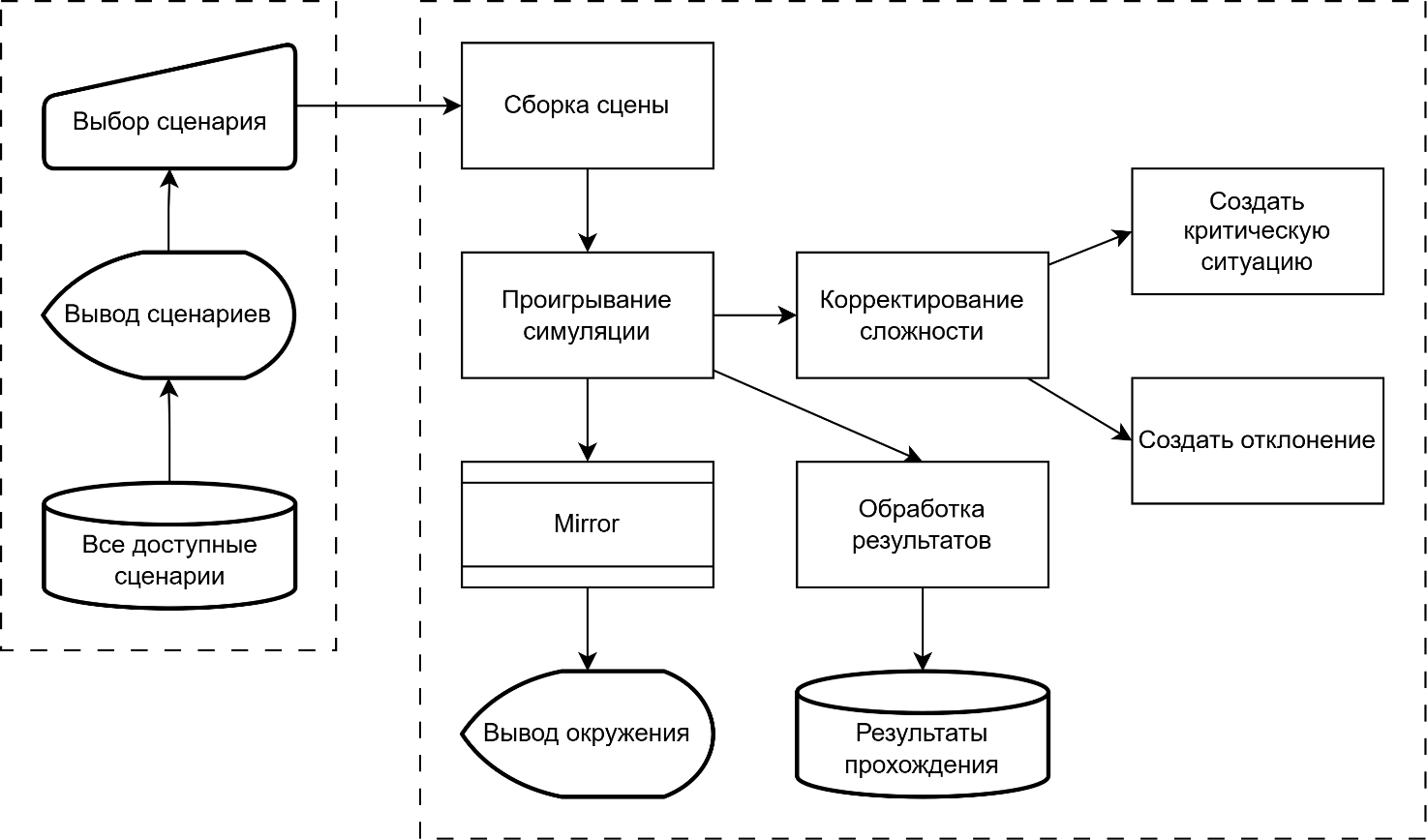


Рис. 12. Функциональная схема тренажёра

Таблица 7

Описание схемы данных

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Тип блока** | **Описание** |
| 1 | 2 | 3 |
| Все доступные сценарии | База данных | База данных всех сценариев для прохождения |
| Вывод сценариев | Дисплей | Вывод всех доступных сценариев для выбора |
| Выбор сценария | Ручной ввод | Выбор сценария симуляции из предложенных |
| Сборка сцены | Процесс | Сборка окружения симуляции из моделей оборудования на основе выбранного сценария |
| Проигрывание симуляции | Процесс | Обновление состояния симуляции |

Продолжение табл. 7

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Действия оператора с VR-гарнитурой | Ручной ввод | Комплекс оборудования для захвата действий оператора |
| Mirror | Библиотека | Библиотека в Unity для синхронизации состояния окружения симуляции между операторами по сети |
| Вывод окружения | Дисплей | Видеопоток симуляции |
| Корректирование сложности | Процесс | Процесс анализа текущего сценария и создания случайных событий, исходя из текущей сложности |
| Создать критическую ситуацию | Подпрограмма | Создание аварий и прочих критических ситуаций на модели рабочего пространства |
| Создать отклонение | Подпрограмма | Создание отклонений у моделей оборудования |
| Обработка результатов | Процесс | Получение результатов симуляции, анализ и подведение итогов прохождения |
| Результаты прохождения | База данных | База данных результатов прохождения симуляций |

## Лабораторная работа №3

## Разработка и оптимизация модульной структуры.

Модульная структура программы представляет собой древовидную структуру, в узлах которой размещаются программные модули, а направленные дуги показывают статическую подчинённость модулей. Если в тексте модуля имеется ссылка на другой модуль, то их на структурной схеме соединяет дуга, которая исходит из первого и входит во второй модуль. Другими словами, каждый модуль может обращаться к подчинённым ему модулям. При этом модульная структура программной системы, кроме структурной схемы, должна включать в себя совокупность спецификаций модулей, образующих эту систему.

Структурные карты Константайна являются моделью отношений иерархии между программными модулями. Узлы структурных карт соответствуют модулям и областям данных, потоки изображают межмодульные вызовы. При этом циклические и условные вызовы модулей моделируются специальными узлами, поэтому потоки должны быть изображены проходящими через эти специальные узлы. Межмодульные связи по данным и управлению также моделируются специальными узлами, привязанными к потокам (т.е. к вызовам модулей), стрелками указываются направления потоков и связей.

Одним из способов оценки качества проекта является анализ сцепления модулей. Сцепление является мерой взаимозависимости модулей. В хорошем проекте сцепления должны быть минимизированы, т.е. модули должны быть слабозависимыми (независимыми) настолько, насколько это возможно. Слабое сцепление между модулями служит признаком хорошо спроектированной системы по следующим причинам:

1. уменьшение количества соединений между двумя модулями приводит к уменьшению вероятности появления “волнового эффекта” (ошибка в одном модуле влияет на работу других модулей);
2. минимизация риска появления “эффекта ряби” (внесение изменений, например, при исправлении ошибки, приводит к появлению новых ошибок), т.к. изменение влияет на минимальное количество модулей;
3. при сопровождении модуля отсутствие необходимости беспокоиться о внутренних деталях других модулей;
4. упрощение системы для понимания, насколько это возможно.

Слабое сцепление может быть достигнуто за счет комбинирования трех следующих способов действий:

1. удаления необязательных связей;
2. уменьшения количества необходимых связей;
3. упрощением необходимых связей.

Практические рекомендации для ослабления сцепления модулей:

1. Создавайте минимальные по количеству параметров межмодульные связи.
2. Создавайте прямые (а не косвенные) межмодульные связи, поскольку интерфейс между двумя модулями наиболее понятен (и, следовательно, менее сложен), если человек может постигнуть его сразу без предварительной ссылки к некоторым другим информационным объектам.
3. Создавайте локализованные связи (например, значения списка параметров вычисляйте непосредственно перед вызовом модуля).
4. Создавайте явные связи. Красноречивым примером неявной связи является взаимодействие модуля А с модулем В за счет модификации области данных из В.
5. Создавайте гибкие связи для облегчения модификаций.

Два модуля А и В являются нормально сцепленными, если:

* А вызывает В,
* В возвращает управление А,
* вся информация, передаваемая между А и В, представляется значениями параметров при вызове.

Существует три типа нормального сцепления: сцепление по данным, сцепление по образцу, сцепление по управлению. На практике наиболее часто используемым типом сцепления является сцепление по данным (data coupling). Два модуля сцеплены по данным, если они взаимодействуют через передачу параметров и при этом каждый параметр является элементарным информационным объектом. В случае небольшого количества передаваемых параметров сцепление по данным обладает наилучшими характеристиками в соответствии с п.п.1-5.

Два модуля сцеплены по образцу (stamp coupling), если один посылает другому составной информационный объект, т.е. объект, имеющий внутреннюю структуру. Примером составного объекта может быть объект Данные о клиенте, включающий в себе поля: Название организации, Почтовый адрес и т.п.

Два модуля сцеплены по управлению (control coupling), если один посылает другому информационный объект - флаг, предназначенный для управления его внутренней логикой. Существует два типа флагов - описательный и управляющий. Описательный флаг обычно описывает ситуацию, которая произошла, и именуются с использованием существительных и прилагательных: Конец файла, Введенная кредитная карта. Управляющий флаг используется для декларирования определенных действий в вызываемом модуле и именуется с использованием глаголов: «Читать следующую запись», «Установить в начало». В общем случае управляющие флаги усиливают сцепление и, следовательно, ухудшают качество проекта.

Как уже отмечалось, вышеперечисленные три типа нормального сцепления в разной степени поддерживают суть модульности и являются приемлемыми в структурном проектировании. Ниже определяются два вида сцепления, которые выходят за пределы хорошей модульности.

Два модуля являются сцепленными по общей области (common coupling), если они ссылаются к одной и той же области глобальных данных. Сцепление по общей области является плохим по следующим причинам.

Во-первых, ошибка в любом модуле, использующем глобальную область, может неожиданно проявить себя в любом другом модуле, также использующем эту глобальную область, поскольку эти глобальные данные не находятся “под защитой” модуля.

Во-вторых, модули, ссылающиеся к глобальным данным, обычно используют точные имена (в отличие от модулей, которые вызываются с использованием параметров). Следовательно, гибкость модулей, сцепленных глобально, намного хуже, чем гибкость нормально сцепленных модулей.

В-третьих, программы с большим количеством глобальных данных чрезвычайно трудны для понимания сопровождающим программистом, поскольку трудно определить, какие данные используются отдельным модулем.

Два модуля являются сцепленными по содержимому (content coupling), если один ссылается внутрь другого любым способом, например, если один модуль передает управление или выполняет переход в другой модуль, если один модуль ссылается (или изменяет) значения информационных объектов в другом модуле или если один модуль изменяет код другого модуля. Такое сцепление делает абсурдной концепцию модулей как черных ящиков, поскольку оно вынуждает один модуль знать о точном содержании и реализации другого модуля [https://dit.isuct.ru/IVT/BOOKS/IS/IS8/defs72.htm].

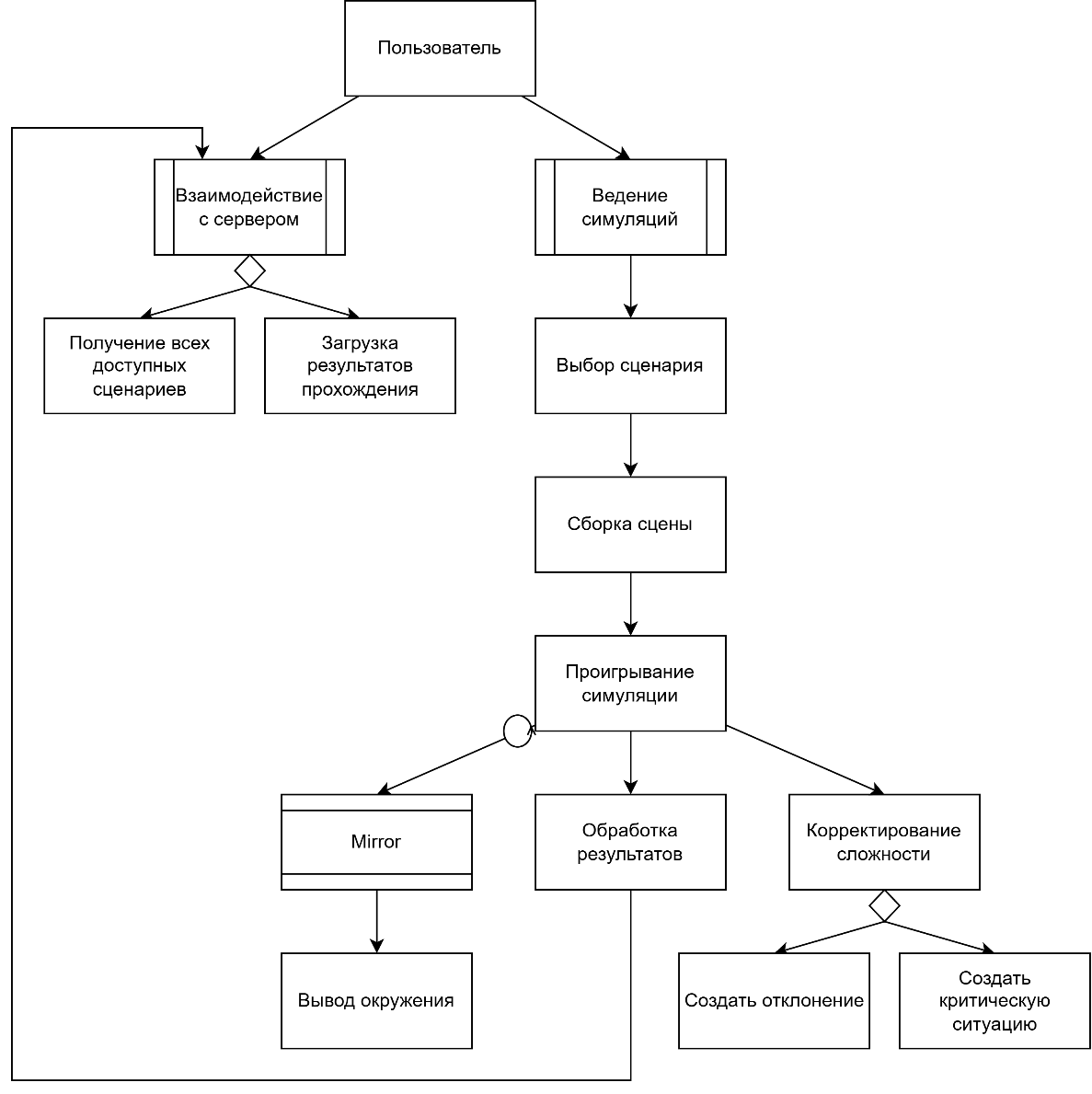


Рис. 13. Модульная структура тренажёра

Оптимизация с целью уменьшения сцепления модулей

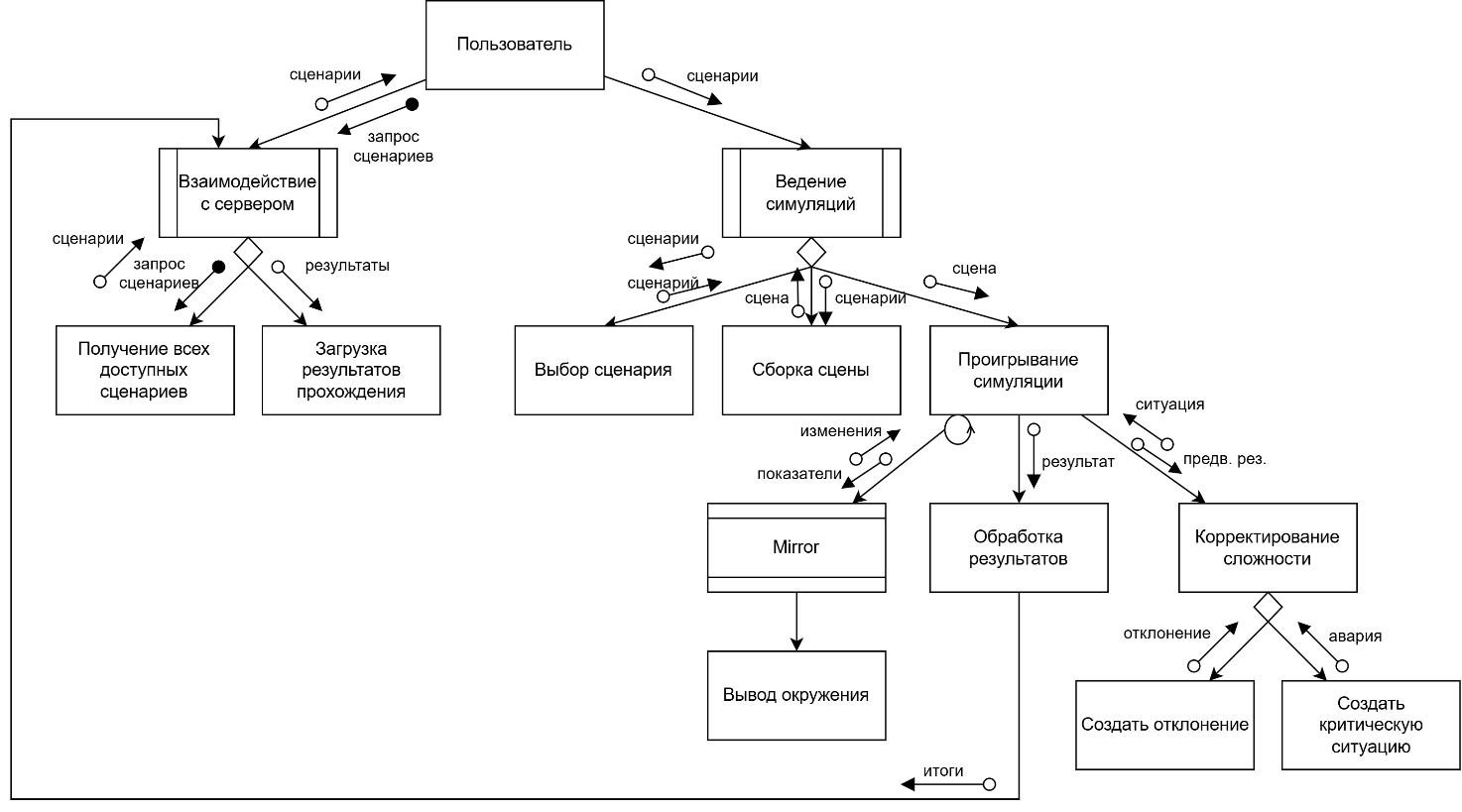


Рис. 14. Карта Константайна тренажёра

Таблица 8

Описание карты Константайна

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Входные данные** | **Выходные данные** | **Описание** |
| 1 | 2 | 3 | 4 |
| Взаимодействие с сервером | Запрос сценариев, Результаты прохождения | Доступные сценарии | Подсистема взаимодействия с базой данных |
| Получение всех доступных сценариев | Запрос всех сценариев | Доступные сценарии | Получить все сценарии |
| Загрузка результатов прохождения | Результаты прохождения | - | База данных результатов прохождения симуляций |
| Ведение симуляции | Сценарии | Сценарии, Сценарий, Сцена | Подсистема ведения симуляции согласно сценарию |
| Выбор сценария | Сценарии | Сценарий | Выбор сценария симуляции из предложенных |

Продолжение табл. 8

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Сборка сцены | Сценарий | Сцена | Сборка окружения симуляции из моделей оборудования на основе выбранного сценария |
| Проигрывание симуляции | Сцена | Показатели, Предварительные результаты, результаты | Модуль обновления состояния и окружения симуляции |
| Mirror | Показатели | Изменения | Библиотека в Unity для синхронизации состояния окружения симуляции между операторами по сети |
| Вывод окружения | - | - | Видеопоток симуляции |
| Корректирование сложности | Предварительные результаты | Ситуация | Процесс анализа текущего сценария и создания случайных событий, исходя из текущей сложности |
| Создать критическую ситуацию | - | Авария | Создание аварий и прочих критических ситуаций на модели рабочего пространства |
| Создать отклонение | - | Отклонение | Создание отклонений у моделей оборудования |
| Обработка результатов | Результаты | Итоги | Получение результатов симуляции, анализ и подведение итогов прохождения |