|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***БАКАЛАВРА НА ТЕМУ:***

***\_\_\_Программная подсистема имитационного моделирования процессов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-83Б |  |  | А.А. Бушев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Руководитель |  |  |  | М.В. Фетисов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Нормоконтролер |  |  |  | Х.Х. ХХХХХХХХХХ |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
|  |  |  |  |  |

*2023 г.*

*ЗАДАНИЕ ЛИСТ 1*

*ЗАДАНИЕ ЛИСТ 2*

*КАЛЕНДАРНЫЙ ПЛАН*

**АННОТАЦИЯ**

Расчетно-пояснительная записка выпускной квалификационной работы бакалавра посвящена процессу проектирования и разработки программной подсистемы имитационного моделирования процессов. Для реализации подсистемы был проведен анализ библиотек и фреймворков поддержки межпроцессного взаимодействия. Помимо этого, был разработан протокол взаимодействия между плагином интеграции со средой разработки и сервером имитационного моделирования.

На основе проведенных исследований была спроектирована и реализована программная подсистема имитационного моделирования процессов.

**ANNOTATION**

Calculation and explanation summary of graduation qualification work of the bachelor is devoted to process of designing and development of software subsystem of process simulation. For subsystem realization analysis of libraries and frameworks supporting interprocess communication is conducted. Also protocol of interaction between the integration plugin with the development environment and the simulation server was developed.

Based on conducted research program, the software subsystem of process simulation was developed.

**РЕФЕРАТ**

Расчетно-пояснительная записка X с., X рис., X табл., X источн., X прил.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, ПОДСИСТЕМА, МЕЖПРОЦЕССНОЕ ВЗАИМОДЕЙСТВИЕ

Объектом разработки является программная подсистема имитационного моделирования процессов.

Целью данной работы является подсистема, которая позволит проводить имитационное моделирование динамических процессов в адаптивной среде разработки.

В ходе выполнения работы были решены следующие задачи:

⎯ исследование библиотек и фреймворков поддержки межпроцессного взаимодействия;

⎯ сравнение аналогов;

⎯ проектирование и разработка диаграмм;

⎯ реализация программной подсистемы.

В результате разработки была спроектирована, реализована и протестирована подсистема, которая предназначена для имитационного моделирования динамических процессов студентами кафедры «Системы автоматического управления» в адаптивной среде разработки SIMODO.

**СОДЕРЖАНИЕ**

олрдлордлрдлордлордлор

родлрлдордлордлрдлордлорд

ордлордлордлрдлордлордлор

олрдлордлрдлордлордлор

родлрлдордлордлрдлордлорд

ордлордлордлрдлордлордлор

олрдлордлрдлордлордлор

родлрлдордлордлрдлордлорд

ордлордлордлрдлордлордлор

олрдлордлрдлордлордлор

родлрлдордлордлрдлордлорд

ордлордлордлрдлордлордлор

олрдлордлрдлордлордлор

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

SIMODO — название программных продуктов, разрабатываемых на кафедре ИУ6 МГТУ им. Баумана.

Git — система контроля версий.

LSP — открытый протокол, разрешающий реализацию и распространение поддержки языка программирования независимо от любого данного редактора или интегрированной среды разработки.

LSP+ — модификация протокола LSP под требования интегрированной среды разработки SIMODO.

ОС — операционная система.

Windows — операционная система для компьютеров от корпорации Microsoft.

Linux — семейство операционных систем на основе открытого ядра.

TCP/IP — сетевая модель передачи данных, представленных в цифровом виде.

UDP — протокол пользовательских датаграмм, один из ключевых элементов набора сетевых протоколов для Интернета.

ПП – программный продукт

ООП – объектно-ориентированное программирование

scriptC0 – первый скриптовый язык SIMODO

ПО – программное обеспечение

RPM – Red Hat Package Manager – формат пакетов программногообеспечения, а также программа, созданная для управления этимипакетами,используемые в ряде Linux-дистрибутивов

DEB – расширение имён файлов «бинарных» пакетов для

распространения и установки программного обеспечения в операционнойсистеме проекта Debian, и других, использующих систему управленияпакетамиdpkg

NSIS – Nullsoft Scriptable Install System – система созданияустановочныхпрограмм для Microsoft Windows с открытым исходным кодомMSI – Microsoft Installer – подсистема Microsoft Windows, обеспечивающая установку программ

QtIFW – Qt Installer Framework – фреймворк для разработкиустановщиков с графическим интерфейсом

DevOps – Методология автоматизации технологических процессовсборки, настройки и развёртывания программного обеспечения

Gitlab – Веб-инструмент жизненного цикла DevOps с открытымисходным кодом

**ВВЕДЕНИЕ**

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

**1 Анализ библиотек и фреймворков поддержки межпроцессного взаимодействия**

В настоящее время для адаптивной системы моделирования SIMODO[1] актуален вопрос межпроцессного взаимодействия. Само моделирование должно происходить отдельно от визуализации результатов моделирования. Однако возникает вопрос, какой инструмент использовать для организации взаимодействия процесса моделирования и процесса визуализации результатов моделирования.

Исследовательская работа посвящена обзору существующих библиотек и фреймворков поддержки межпроцессного взаимодействия. А также проведению сравнительного анализа библиотек и фреймворков поддержки межпроцессного взаимодействия. Помимо этого, в работе разработан протокол взаимодействия с адаптивной системой моделирования SIMODO.

**1.1 Описание открытой архитектуры интегрированной среды разработки SIMODO**

Основной задачей интегрированная среды разработки SIMODO является обеспечение удобного средства описания моделей (помимо запуска моделирования и анализа результатов моделирования) на базе открытой архитектуры[2]. Открытая архитектура интегрированной среды разработки SIMODO позволяет расширить базовую функциональность для работы с различными документами.

Основной задачей интегрированной среды разработки SIMODO является предоставление простого средства редактирования текста моделей. Поскольку любой язык программирования может быть использован для написания фрагментов модели, то среда предоставляет гибкие средства настройки, не замыкаясь на наборе предметно-ориентированных языков SIMODO.

На рисунке 1 изображён графический интерфейс интегрированной среды разработки SIMODO.

Интегрированная среда разработки SIMODO состоит из расширяемого редактора — оболочка редактора и расширения (плагины) — и процессов, выполняющие разнообразные функции, которые целесообразно отделить от редактора межпроцессным взаимодействием[1]. Преимуществом такого отделения является отказоустойчивость всей интегрированной среды разработки SIMODO.

Оболочка редактора предоставляет базовую функциональность и механизм расширения. После запуска оболочка редактора загружает плагины. Плагины являются фабриками для соответствующих компонентов. Компоненты, которые поставляются плагинами, расширяют функциональность редактора. В свою очередь оболочка редактора предоставляет для указанных плагинов и порождающих ими компонентов интерфейс доступа к некоторым своим функциям.

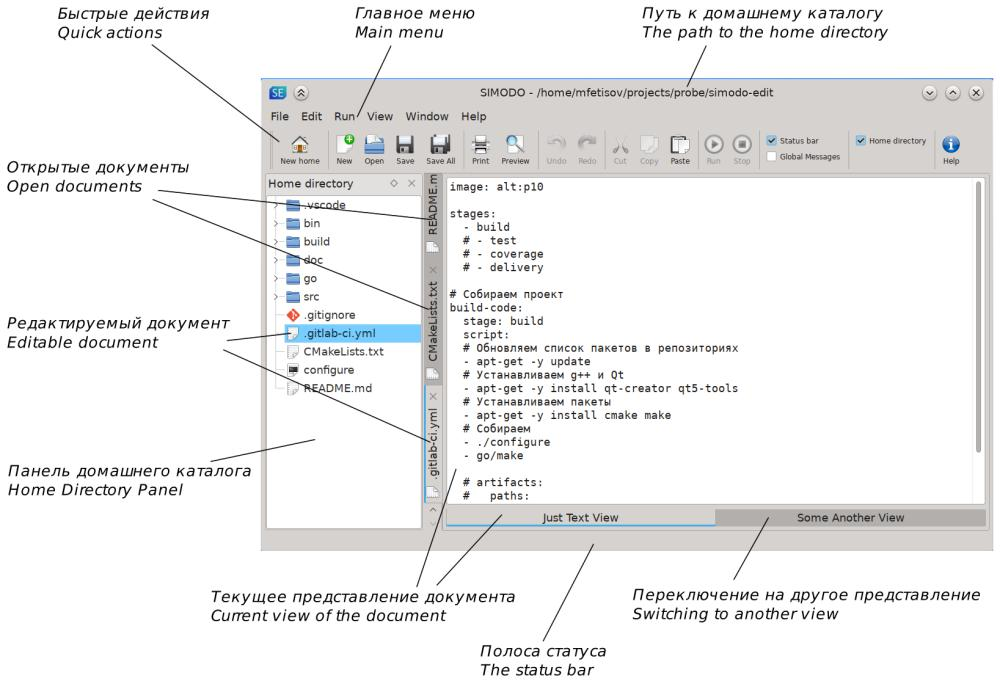


Рисунок 1 — Графический интерфейс интегрированной среды разработки SIMODO.

Сама оболочка не выполняет отображение или управление документом. Вместо этого она предоставляет пространство, где размещает панели и представление документов, отображение которых определяется реализацией плагинов.

На рисунке 2 представлена схема интерфейсов оболочки.

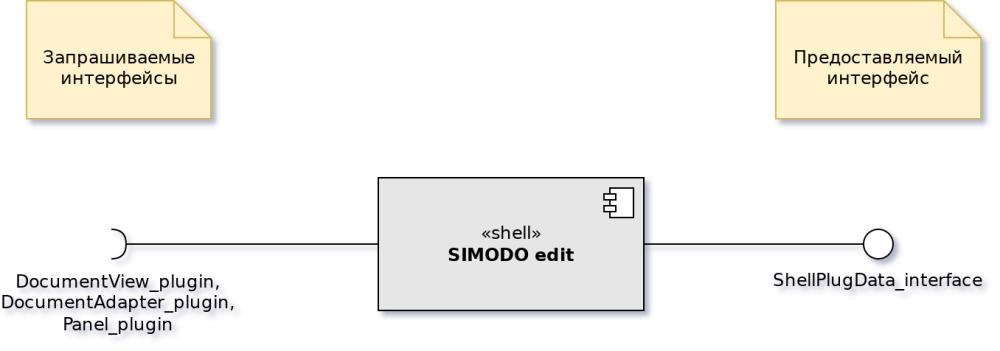


Рисунок 2 — Интерфейсы оболочки

Следующим слоем открытой программной архитектуры являются процессы, выполняющие разнообразные функции, которые целесообразно отделить от редактора межпроцессным взаимодействием. Преимуществом такого отделения является отказоустойчивость всей программы.

Процессы запускаются конкретными плагинами как консольные приложения и обмениваются между ними через каналы — механизм взаимодействия через стандартный ввод-вывод. Возможен как синхронный, так асинхронный обмен.

Примерами выделения в отдельные процессы могут быть: взаимодействие с системами управления версиями (например, git), языковые протоколы (например, LSP[3]; для языков серии SIMODO используется модифицированная версия LSP+[1]), сервер моделирования SIMODO и т.д.

На рисунке 3 представлено взаимодействия слоёв интегрированной среды разработки SIMODO.

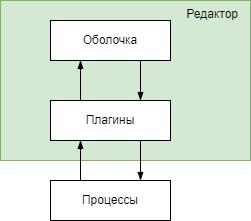


Рисунок 3 — Взаимодействие элементов интегрированной среды разработки SIMODO

Открытая программная архитектура интегрированной среды разработки SIMODO представляет собой вариант многоуровневой архитектуры, где нижележащие слои не зависят от вышележащих, а обмен сообщениями возможет только между соседними слоями.

Адаптивная система моделирования SIMODO относится к третьему слою архитектуры интегрированной среды разработки SIMODO. Среда должна выполняться в отдельном процессе, с которым плагин расширяемого редактора обменивается сообщениями.

Начало и завершение работы адаптивной системы моделирования SIMODO, а также управление ходом моделирования должны программно контролироваться плагином расширяемого редактора.

Таким образом возникает необходимость выбора инструмента реализации межпроцессного взаимодействия между плагинами и процессами.

**1.2 Сравнительный анализ существующих библиотек и фреймворков поддержки межпроцессного взаимодействия**

Для выявления достоинств и недостатков существующих библиотек и фреймворков, необходимо провести их сравнительный анализ, на основе которого будет выполнен выбор библиотеки или фреймворка.

Исходя из того, что в качестве целевой операционной системы для интегрированной среды разработки SIMODO является ОС Windows и дистрибутивы Linux[1], при выборе аналогов, будут рассматриваться кросс- платформенные библиотеки и фреймворки, которые разработаны под указанные ОС.

На основании того, что в качестве основного языка разработки интегрированной среды разработки SIMODO используется язык программирования C++[1], при выборе аналогов, будут рассматриваться библиотеки и фреймворки, которые обладают интерфейсом, совместимым с языком программирования C++.

На сегодняшний день существует множество библиотек и фреймворков поддержки межпроцессного взаимодействия. Одними из самых популярных являются [4]:

— Boost;

— eCAL;

— ACE;

— POCO;

— Qt.

**Boost**

Библиотека Boost.Process позволяет управлять системными процессами. Она предоставляет следующие возможности[5]:

— Создание дочернего процесса.

— Настройка каналов для дочернего процесса.

— Синхронное и асинхронное взаимодействие с дочерним процессом через каналы.

— Синхронное и асинхронное ожидание завершения процесса.

— Прерывание процесса.

Библиотека Boost.Interprocess снижает сложность использования межпроцессного взаимодействия и синхронизации, а также предлагает ряд механизмов межпроцессного взаимодействия и синхронизации[5]:

— разделяемая память;

— файлы, проецируемые в память;

— cемафоры;

— мьютексы;

— условные переменные;

— размещаемые в разделяемой памяти и файлах, проецируемых в память, мьютексы;

— именованные объекты синхронизации;

— блокировка файлов;

— относительные указатели;

— очередь сообщений.

Библиотека Boost.Interprocess также предлагает высокоуровневые межпроцессного механизмы динамического выделения разделяемой памяти или файлов, проецируемых в память. Библиотека не требует компиляции, потому что состоит только из заголовочных файлов.

**eCAL**

Фреймворк eCAL (улучшенный абстрактный уровень связи) является связующим программным обеспечением, которое делает возможным масштабируемое, высокопроизводительное межпроцессное взаимодействие на одном вычислительном узле или между несколькими узлами в вычислительной сети[6]. eCAL разработан для характерных сценариев облачных вычислений, когда разные процессы обмениваются данными используя шаблон издатель-подписчик.

Фреймворк eCAL автоматически выбирает лучший механизм передачи данных для каждого соединения:

— разделяемая для связи на одном вычислительном узле;

— UDP протокол для сетевого взаимодействия.

На рисунке 4 показан принцип работы фрейморка eCAL.

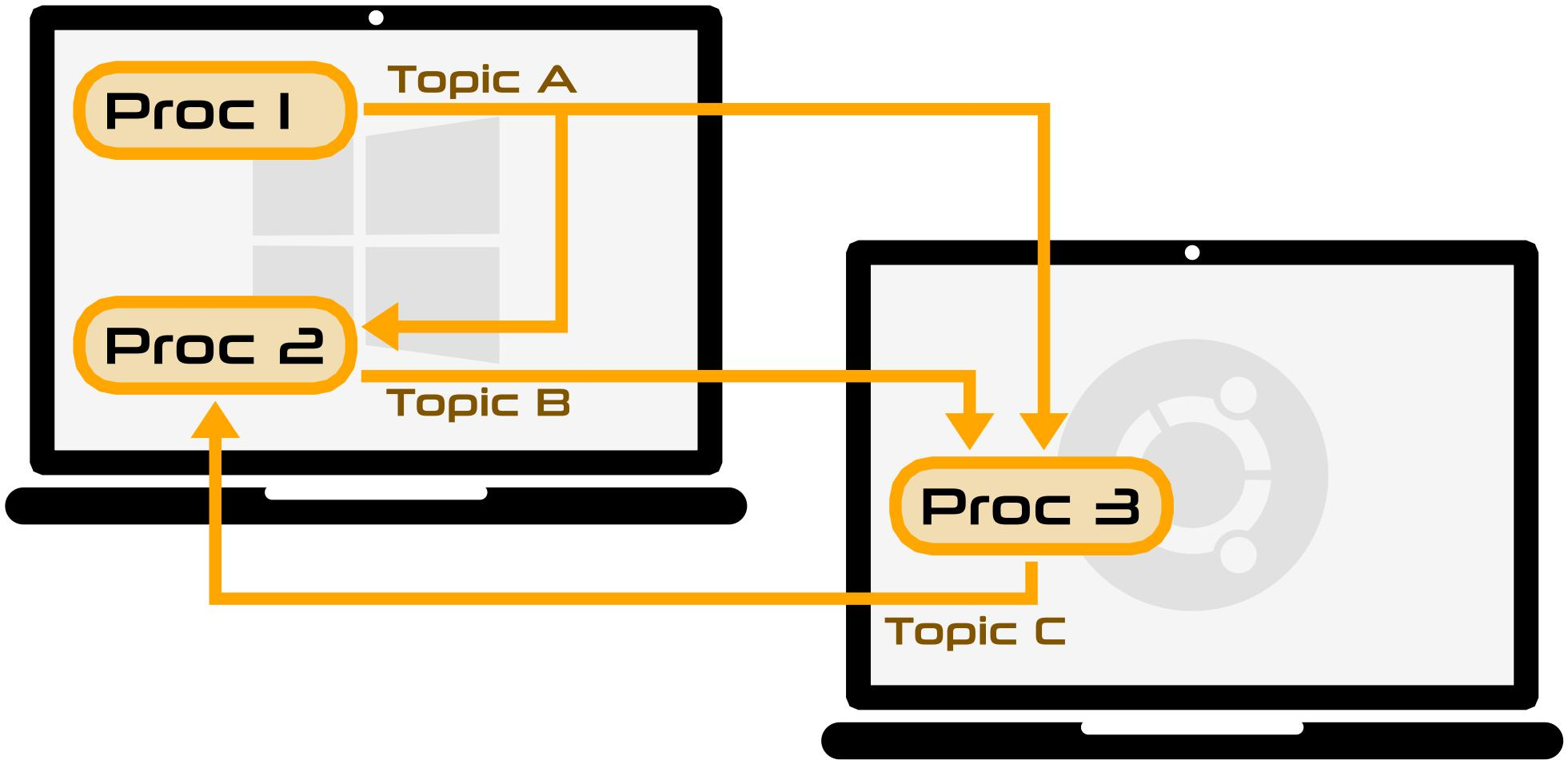


Рисунок 4 — Принцип работы фреймворка eCAL

Фреймворк eCAL обладает следующими особенностями[6]:

— скорость обмена данными 1–10 Гб/с;

— поддержка шаблона издатель-подписчик и сервер-клиент;

— обмен данных происходит без посредника;

— обладает мощным инструментарием отслеживания и анализа децентрализованных потоков данных;

— не требует настройки перед использование;

— позволяет использовать пользовательский протокол сообщений;

— работает на ОС Windows и Linux.

**ACE**

Фреймворк ACE (адаптивное окружение связи) является объектно- ориентированным фреймворком, который реализует основные шаблоны для программного обеспечения параллельной связи. Фреймворк ACE предоставляет широкий набор C++ фасадов-обёрток, предназначенных для многократного использования, и компонент фреймворка, выполняющих общие для ряда ОС задачи программной связи[7]:

— Демультиплексирование событий и диспетчеризация обработки событий.

— Обработка сигналов.

— Инициализация сервисов.

— Межпроцессное взаимодействие.

— Управление разделяемой памятью.

— Маршрутизация сообщений.

— Динамическая настройка распределённых сервисов.

— Параллельное выполнение и синхронизация.

Фреймворк ACE предназначен для разработчиков высокопроизводительных сервисов связи и приложений реального времени[7]. Фреймворк упрощает разработку объектно-ориентированных сетевых приложений и сервисов, использующие межпроцессное взаимодействие, демультиплексирование событий, явное динамическое связывание и параллелизм. В дополнение, Фреймворк ACE автоматизирует системную настройку с помощью сервисов динамического связывания во время выполнения приложений и выполняет данные сервисы в одном или нескольких процессах или потоках.

Диаграмма на рисунке 5 иллюстрирует ключевые компоненты в фреймворке ACE и их иерархические связи.

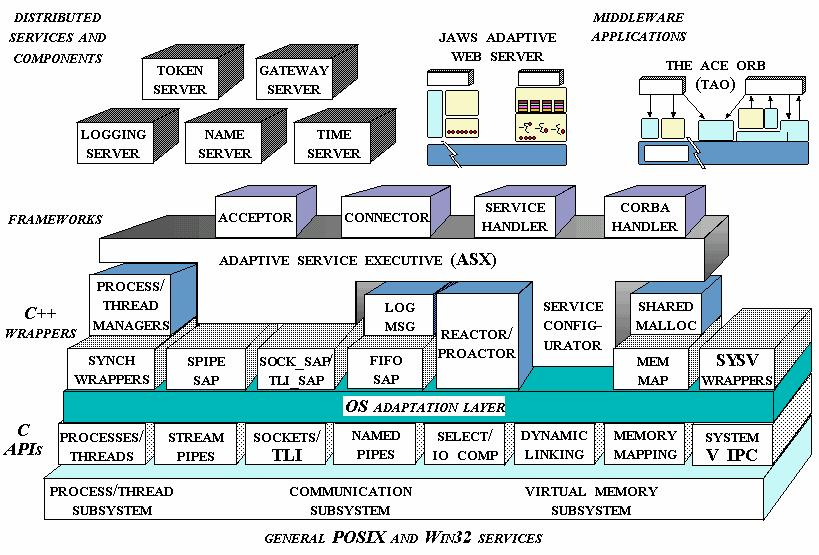


Рисунок 5 — Структура и функциональность фреймворка ACE

**POCO**

Библиотеки POCO являются кросс-платформенными библиотеками для создания ориентированных на сеть или интернет приложений, которые работают на персональных компьютерах, серверах, мобильных устройствах, устройствах интернета вещей и встроенных системах[8].

Библиотека POCO Process позволяет:

— Получить информацию о текущем процессе.

— Запустить новый процесс.

— Прервать другой процесс.

— Перенаправить ввод-вывод запускаемого процесса.

— Синхронизировать процессы, используя примитивы синхронизации: именованный мьютекс и именованное событие.

— Использовать разделяемую память.

Библиотека POCO является кросс-платформенной, модульной и масштабируемой библиотекой, используемой в программах корпоративного уровня[8]. Библиотека обеспечивает целостный и простой для понимания программный интерфейс.

Исходные тексты библиотеки POCO обладают особым качеством кода, читаемостью, полнотой, согласованностью и тестируемостью.

Состав библиотеки POCO представлен на рисунке 6.

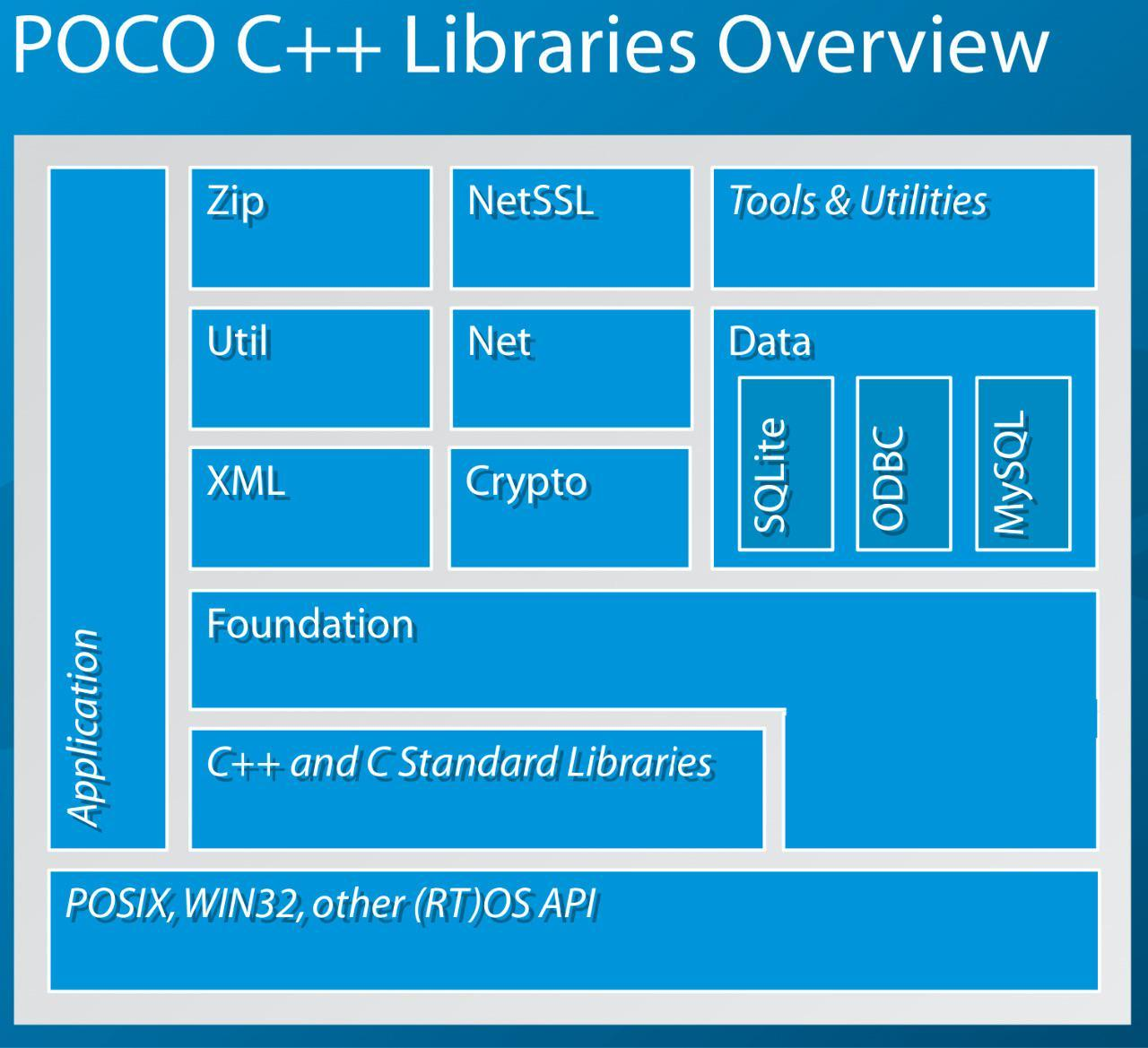


Рисунок 6 — Состав библиотеки POCO.

**Qt**

Фреймворк Qt является кросс-платформенным программным обеспечением для создания как графических пользовательских интерфейсов, так и кросс-платформенных приложений, запускаемых на различном программном и аппаратном обеспечении, например, операционные системы Linux, Windows, macOS, Android или встраемые системы с минимальными изменениями нижележащей кодовой базы, оставаясь нативным приложением с соответствующими возможностями и скоростью[9].

Фреймворк Qt предоставляет несколько кросс-платформенных способов реализации межпроцессного взаимодействия[10]:

— Использование сетевого TCP/IP межпроцессного взаимодействия в вычислительной сети.

— Использование локального сервера и сокетов для межпроцессного взаимодействия на одном вычислительном узле.

— Разделяемая память.

— Класс QProcess позволяет запускать внешние программы в качестве дочерних процессов и взаимодействовать с ними.

— Управление сессиями, позволяющее распространять события между процессами.

Терминология для каналов в фреймворке Qt может ввести в заблуждение[10]. Выходные каналы процесса являются каналами чтения, а входные каналы процесса являются каналами записи. Данная терминология обусловлена тем, что из процесса читается результат его работы, а запись в процесс становится его входом.

Фреймворк Qt позволяет перенаправить только один выходной канал.

Интегрированная среда разработки SIMODO разрабатывается с использованием фреймворка Qt[1], поэтому механизм плагинов реализуется с использованием данного фреймворка. Каждый плагин, расширяющий функциональность оболочки будет зависеть от небольшой части фреймворка Qt, связанного с системой плагинов.

Состав фреймворка Qt представлен на рисунке 7.

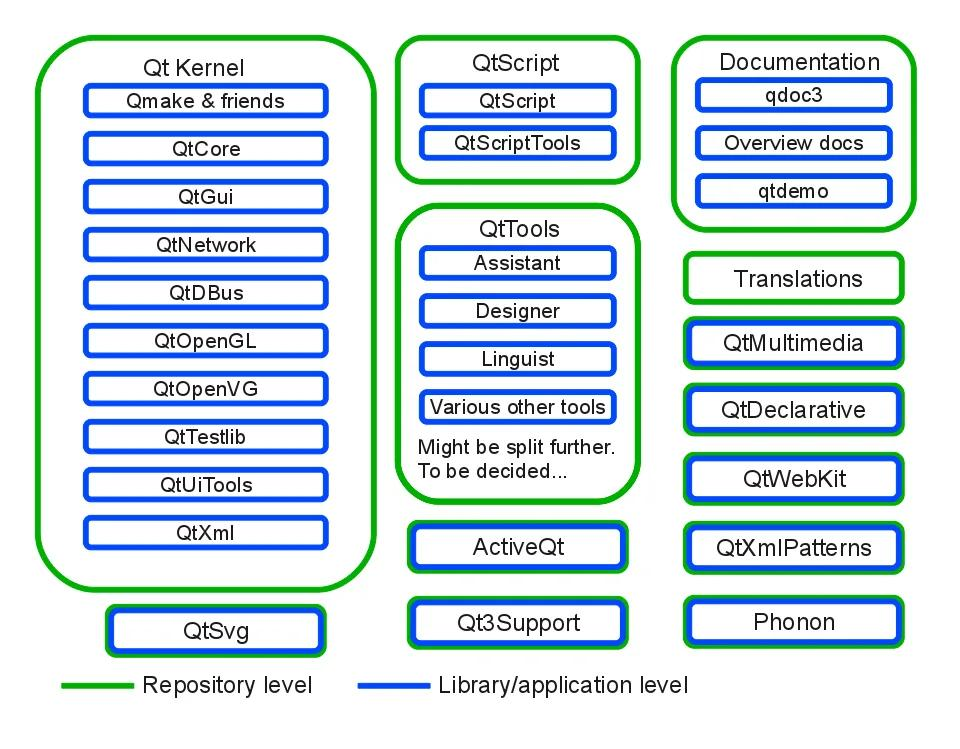


Рисунок 7 — Состав фреймворка Qt.

Анализа данных библиотек и фреймворков показывает, что инструменты реализации межпроцессного взаимодействия предоставляют широкий выбор методов межпроцессного взаимодействия. Однако для поставленной задачи нет необходимость в использовании всего инструментария тяжеловесных библиотек. Выделим критерии выбора:

— Возможность программного создания процесса: интегрированная среда разработки должна самостоятельно запускать сторонние процессы, чтобы этого не приходилось делать пользователю.

— Взаимодействие с процессом через каналы[11]: каналы являются наиболее простым способом взаимодействия с процессом, что позволит ограничить работу адаптивной системы моделирования SIMODO с вводом-выводом до работы со стандартными потоками ввода-вывода.

— Программное завершение процесса: интегрированная среда разработки SIMODO должна иметь возможность преждевременно завершить работу внешних процессов, не дожидаясь завершения работы этих процессов в случае завершения работы среды.

— Библиотека или фреймворк должен состоять только из заголовочных файлов C++, чтобы исключить издержки, связанные с поставкой правильной версией библиотек времени выполнения программы.

— Размер полностью скомпилированной библиотеки должен быть минимален, чтобы итоговый размер приложения занимал как можно места в постоянном запоминающем устройстве пользователя.

По выделенным критериям проведён сравнительный анализ библиотек, который продемонстрирован в таблице 1

Таблица 1 — сравнительный анализ библиотек поддержки межпроцессного взаимодействия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий \ Библиотека | Программ- ное создание процесса | Взаимо- действие через каналы | Программ- ное прерывание процесса | Состоит только из заголо- вочных файлов | Размер библиотеки, Мб |
| Boost | + | + | + | + | 3 |
| Qt | + | + | + | – | 15 |
| POCO | + | + | + | – | 8 |
| ACE | + | – | + | – | 89 |
| eCal | – | – | – | – | 112 |

Сравнение приложений по выделенным критериям выявило, что среди предложенных библиотек и фреймворков поддержки межпроцессного взаимодействия библиотека. Библиотека Boost обладает самыми лучшими показателями относительно остальных кандидатов.

**1.3 Протокол взаимодействия с адаптивной системой моделирования**

Архитектура интегрированной среды разработки SIMODO предполагает три уровня:

1. Расширяемый редактор — оболочка, задача которой состоит в обеспечении механизма расширения и базовой функциональности (например, отображение окна программы и базовые действия работы с файлами).
2. Плагины — расширения оболочки, которые обеспечивают основную функциональность (например, возможность редактировать текстовые файлы или работать с файловой системой).
3. Процессы — внешние процессы, выполняющие разнообразные функции (например лексический, синтаксический и семантический анализ или выполнение моделирования), которые разумно изолировать от оболочки межпроцессным взаимодействием.

На рисунке 8 приведён пример представления каждого слоя архитектуры интегрированной среды разработки SIMODO.

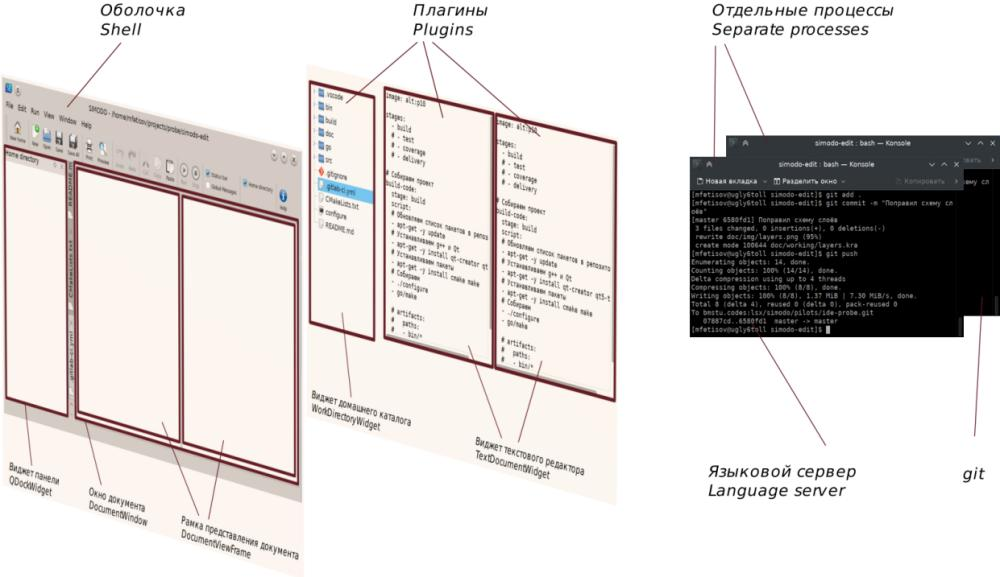


Рисунок 8 — Архитектура интегрированной среды разработки SIMODO

В архитектуре интегрированной среды разработки SIMODO нижние уровни независимы от верхних, а взаимодействие возможно только между смежными уровнями.

Адаптивная система моделирования SIMODO принадлежит к третьему уровню в архитектуре интегрированной среды разработки SIMODO. Среда выполняется в отдельном процессе, с которым взаимодействует плагин расширяемого редактора для передачи исходных текстов моделей, скриптов запуска моделирования или других программных модулей. В свою очередь, среда отправляет информацию о состоянии моделей и ходе моделирования. Среде может понадобится дополнительная информация (например, модель использует внешние модули), поэтому она может запросить дополнительные ресурсы у плагина (например, исходные тексты внешних модулей).

Запуск и остановка адаптивной системы моделирования SIMODO, запуск, временная и полная остановка моделирования должны программно контролироваться плагином расширяемого редактора.

Таким образом, можно выделить следующие сообщения, передаваемые между плагином расширяемого редактора и процессом адаптивной системы моделирования SIMODO:

— init: инициализация адаптивной системы моделирования SIMODO.

— info: запрос информации о текущем состояния адаптивной системы моделирования SIMODO

— start: запуск моделирования с передачей исходного текста сценария, описывающего ход моделирования.

— request: запрос дополнительного ресурса системой моделирования.

— process: передача исходного текста модуля.

— send: передача данных.

— pause: временная остановка моделирования.

— resume: продолжение моделирования.

— stop: полная остановка моделирования.

— terminate: остановка адаптивной системы моделирования SIMODO

На рисунке 9 изображена диаграмма последовательности сообщений между плагином и адаптивной системой моделирования SIMODO (сервером) при нормальном ходе моделирования, т.е. без возникновения ошибок при обмене сообщениями.

Основная концепция взаимодействия между процессом плагина и процессом сервера моделирования заключается в том, что после запуска процесса, на котором работает сервер моделирования, плагин инициирует запуск, управление и остановку моделирования, в свою очередь предоставляя серверу необходимые ресурсы.

В первую очередь плагин запускает процесс сервера моделирования, после чего плагин должен отправить сообщение инициализации серверу. О готовности сервера плагин может узнать отправив запрос на получение информации о состоянии сервера. В ответ на запрос сервер отправит запрашиваемую информацию стандартным сообщением данных.

Моделирование запускается на инициализированном сервере моделирования сообщением start и передаётся сценарий моделирования. После этого нормальным считается следующее поведение:

— Запрос сервером моделирования дополнительных ресурсов и отправка соответствующих ресурсов плагином.

— Отправка плагину данных о ходе моделирования.

— Отправка серверу моделирования команд управления, которые обрабатываются в соответствии со сценарием моделирования.

— Временная приостановка хода моделирования и возобновления хода моделирования.

Завершение хода моделирования возможно в соответствии с логикой сценария моделирования (не приведено на рисунке 9) или при получении сообщения stop.

Сервер моделирования позволяет проводить моделирование неограниченное число раз в общем случае с разными сценариями моделирования

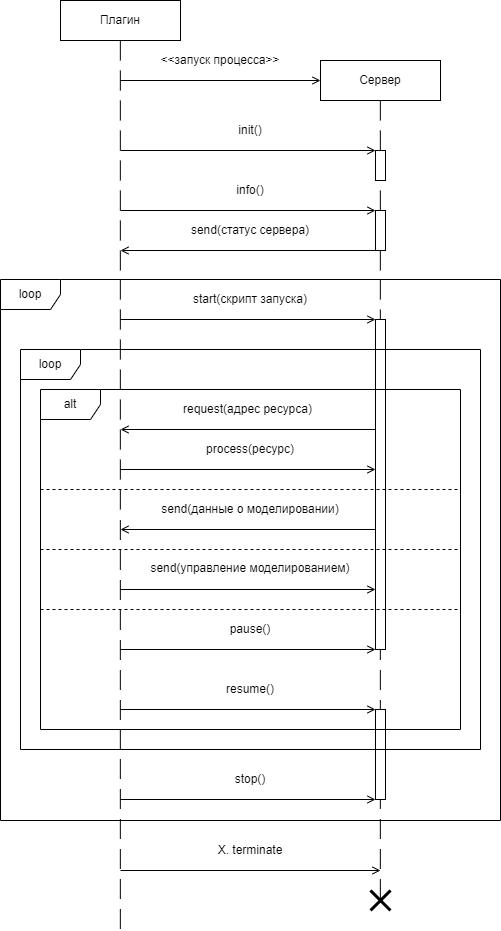


Рисунок 9 — Пример нормального хода моделирования

**1.4 Выводы**

Сравнительный анализ библиотек и фреймворков поддержки межпроцессного взаимодействия, показал, что наиболее подходящей библиотекой для организации межпроцессного взаимодействия между плагином интегрированной среды разработки SIMODO и сервером адаптивной системы моделирования SIMODO является библиотека Boost.

На основе описания открытой архитектуры интегрированной среды разработки SIMODO разработан протокол взаимодействия с адаптивной системой моделирования SIMODO.

**2 Разработка программной подсистемы имитационного моделирования процессов**

**2.1 Выбор технологии и языка программирования**

В качестве технологии программирования было выбрано объектно- ориентированное программирование. ООП является основой всех современных приложений и имеет удобное и практическое применение. При использовании этого метода вся программа разбивается на объекты, с каждым из которых работа происходит по отдельности, что позволяет в будущем расширять программный продукт путем добавления новых объектов.

Для написания настольных приложений существует множество языков программирования. Наиболее популярные[11] – это С#, C++ и Python, которые являются кроссплатформенными языками.

Для разрабатываемого приложения был выбран язык C++, так как он используется в разработке адаптивной среды разработки SIMODO[12].

**2.2 Выбор подхода разработки**

В качестве жизненного цикла разработки была выбрана спиральная модель. Этот метод позволяет в конце каждого цикла иметь работающий продукт, который можно продемонстрировать[13]. Это дает возможность своевременно оценить и протестировать продукт, чтобы сразу вносить какие- либо правки и исправлять ошибки, не дожидаясь окончания разработки. Нахождение багов на каждом этапе позволяет избежать «волнового» исправления ошибок. А также этот метод позволяет детальнее подойти к каждому этапу разработки по отдельности

Также при разработке было решено использовать нисходящий подход[14], реализуя сначала модули верхнего уровня (интерфейс пользователя), а после переходя к модулям нижнего уровням (логика работы программы).

**2.3 Разработка схемы структурной информационной системы**

Структурная схема программного продукта показывает разделение программы на её главные составляющие. На основе анализа технического задания, в разрабатываемом приложении, которое выполняет имитационное моделирование, выявлено две подсистемы:

— подсистема интеграции с адаптивной средой разработки: подсистема интеграции с адаптивной средой разработки преобразует входные данные в понятный для сервера моделирования вид и преобразует исходящие от сервера моделирования данные в приемлемый для среды разработки вид; включает в себя две подсистемы:

1) в подсистеме ввода вывода происходит работа с процессом, в котором работает сервер моделирования;

2) в подсистеме обработки сообщений происходит обработка сообщений, приходящих от сервера;

— подсистема имитационного моделирования включает в себя три подсистемы:

1) в подсистеме ввода вывода происходит получение и отправка данных в рамках протокола сервера моделирования;

2) в подсистеме обработки сообщений происходит обработка сообщений, приходящих от редактора;

3) подсистема интерпретации отвечает за процесс имитационного моделирования в рамках интерпретатора scriptC0; включает в себя три подсистемы:

а) интерпретатор отвечает за исполнение скриптов на языке scriptC0

б) менеджер модулей обеспечивает интерпретатор исходными текстами модулей, запрашиваемых в процессе интерпретации

в) модуль сцены является модулем scriptC0; отвечает за расчёт моделей в виде систем дифференциальных уравнений с помощью численных методов.

На основе выявленных подсистем была составлена структурная схема ПП, показанная на рисунке 10.

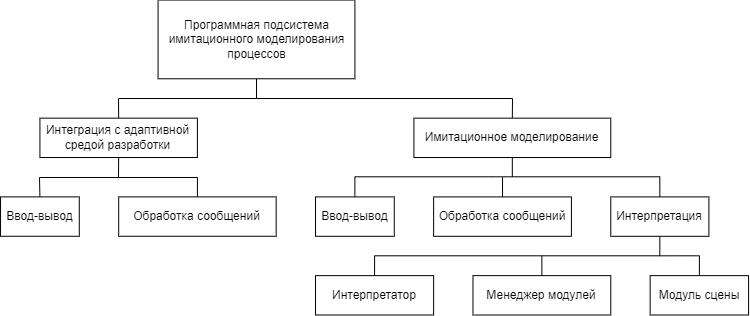


Рисунок 10 – Структурная схема информационной системы

**2.4 Разработка диаграмм последовательностей**

Сервер имитационного моделирования принадлежит к третьему уровню в архитектуре интегрированной среды разработки SIMODO. Сервер выполняется в отдельном процессе, с которым взаимодействует плагин расширяемого редактора для передачи исходных текстов моделей, скриптов запуска моделирования или других программных модулей. В свою очередь, сервер отправляет информацию о состоянии моделей и ходе моделирования. Серверу может понадобится дополнительная информация (например, модель использует внешние модули), поэтому он может запросить дополнительные ресурсы у плагина (например, исходные тексты внешних модулей). Запуск и остановка сервера имитационного моделирования SIMODO, запуск, временная и полная остановка моделирования должны программно контролироваться плагином расширяемого редактора.

Таким образом, можно выделить следующие сообщения, передаваемые между плагином расширяемого редактора и процессом сервера имитационного моделирования SIMODO:

— Инициализация с грамматиками языков программирования в качестве аргументов: инициализация сервера имитационного моделирования SIMODO с предоставление исходных текстов грамматик языков программирования, используемых во время имитационного моделирования.

— Запуск моделирования с основным сценарием в качестве аргумента: запуск интерпретации исходного текста основного сценария моделирования

— Запрос внешнего ресурса по его адресу: сигнал адаптивной среде разработке о необходимости в исходном тексте модуля, который указан в основном сценарии моделирования или в уже полученных модулях.

— Отправка исходного текста ресурса: передача серверу имитационного моделирования исходного текста модуля, который был запрошен по определённому адресу.

— Установка пары ключ-значение: передача информации серверу имитационного моделирования в процессе имитационного моделирования для управления ходом имитационного моделирования.

— Сообщение информации: передача информации адаптивной среде разработки о ходе имитационного моделирования.

— Приостановка моделирования: сигнал серверу имитационного моделирования об обратимом прерывании хода имитационного моделирования.

— Возобновление моделирования: сигнал серверу имитационного моделирования о возобновлении хода имитационного моделирования.

— Остановка: сигнал серверу имитационного моделирования о необратимом прерывании хода имитационного моделирования.

— Остановка сервера: сигнал серверу имитационного моделирования о прерывании работы сервера имитационного моделирования; требуется ожидание момента завершения процесса, в котором исполняется сервер имитационного моделирования.

На рисунке 11 изображена диаграмма последовательности[15] сообщений между плагином и сервером имитационного моделирования SIMODO при нормальном ходе моделирования, т.е. без возникновения ошибок при обмене сообщениями.

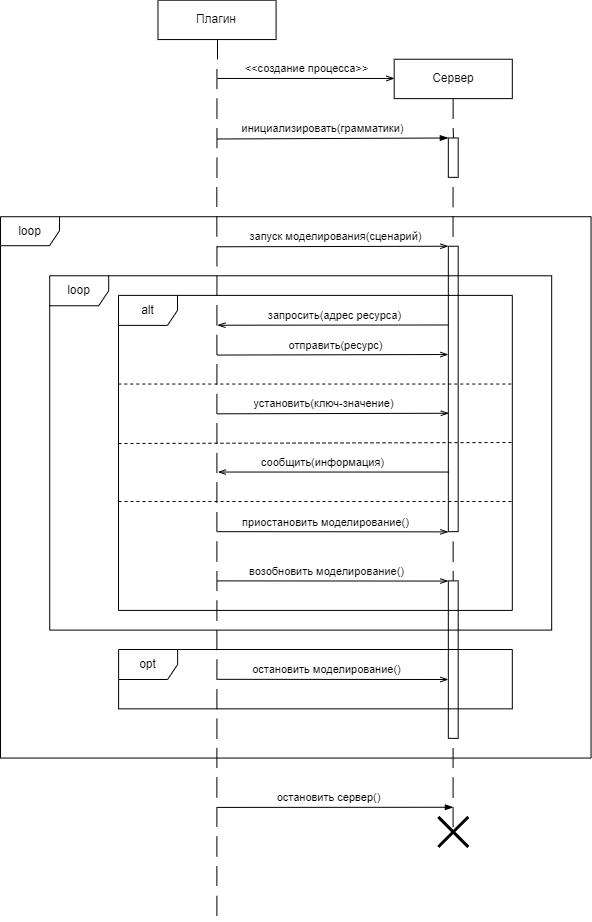


Рисунок 11 – Диаграммы последовательностей

Основная концепция взаимодействия между процессом плагина и процессом сервера моделирования заключается в том, что после запуска процесса, на котором работает сервер моделирования, плагин инициирует запуск, управление и остановку моделирования, в свою очередь предоставляя серверу необходимые ресурсы.

В первую очередь плагин запускает процесс сервера моделирования, после чего плагин должен отправить сообщение инициализации серверу. О готовности сервера плагин может узнать отправив запрос на получение информации о состоянии сервера. В ответ на запрос сервер отправит запрашиваемую информацию стандартным сообщением данных.

Моделирование запускается на инициализированном сервере моделирования и передаётся сценарий моделирования. После этого нормальным считается следующее поведение:

— Запрос сервером моделирования дополнительных ресурсов и отправка соответствующих ресурсов плагином.

— Отправка плагину данных о ходе моделирования.

— Отправка серверу моделирования команд управления, которые обрабатываются в соответствии со сценарием моделирования.

— Временная приостановка хода моделирования и возобновления хода моделирования. Завершение хода моделирования возможно в соответствии с логикой сценария моделирования (не приведено на рисунке 11) или при получении сообщения остановки моделирования.

Сервер моделирования позволяет проводить моделирование неограниченное число раз в общем случае с разными сценариями моделирования.

**2.5 Разработка диаграмм классов предметной области**

После окончания разработки ПО была разработана диаграмма классов предметной области, чтобы представить созданные классы с их полями и методами. Диаграмма классов[16] сервера имитационного моделирования показана на рисунке 12. Диаграмма классов модуля сцены показана на рисунке 13.

Основные классы сервера имитационного моделирования:

— Сервер — класс, организующий конфигурацию сервера имитационного моделирования.

— СервисВводаВывода — интерфейс, обеспечивающий получение и отправку сообщений через абстрактный канал.

— ОбработчикСобытий — интерфейс, позволяющий наращивать функционал сервера созданием нового объекта.

— Интерпретатор — интерфейс, обеспечивающий ход имитационного моделирования.

Основные классы модуля сцены:

— Сцена — класс, отвечающий за управление имитационным моделированием и обеспечением режима реального времени, если он активирован.

— Актор — интерфейс участника сцены; позволяет потенциально использовать любые модели.

— Модель — класс, отвечающий за представление системы дифференциальных уравнений в понятном для сцене виде.

— Композиция — класс, отвечающий за объединение моделей в единицу, которую можно трактовать как отдельную модель; позволяет выстраивать иерархическую структуру сложных моделей.

**2.6 Разработка схемы алгоритмов модуля сцены**

Одной из основных подсистем приложения является модуль сцены. Алгоритм работы модуля сцены необходимо разработать так, чтобы обеспечить пересчёт всех акторов, и если активирован режим реального времени, выполнить пересчёт акторов с адекватной реальному времени задержкой.

Схемы алгоритмов[15] модуля сцены представлены на рисунках 14–15.

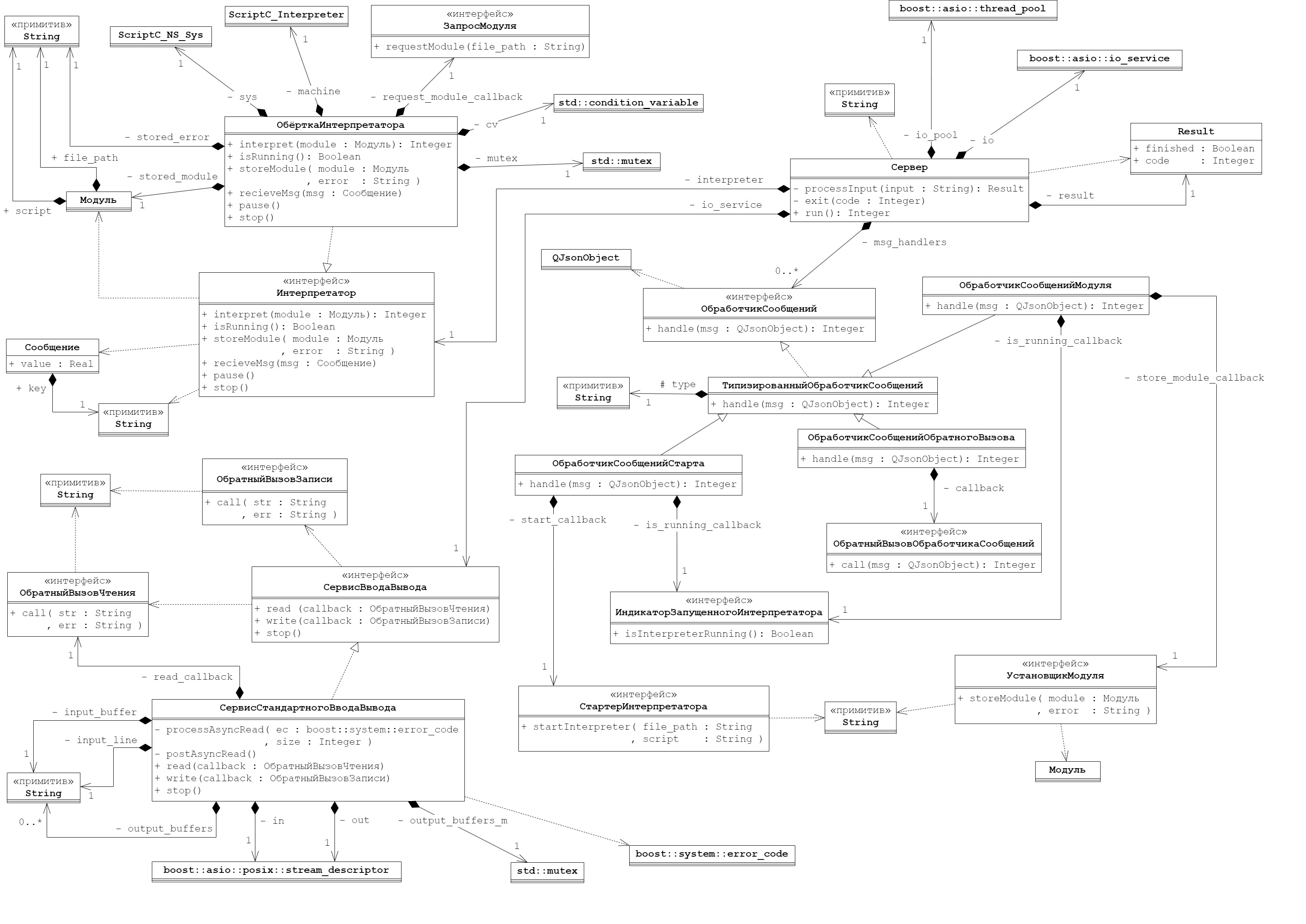


Рисунок 12 – Диаграммы классов сервера имитационного моделирования

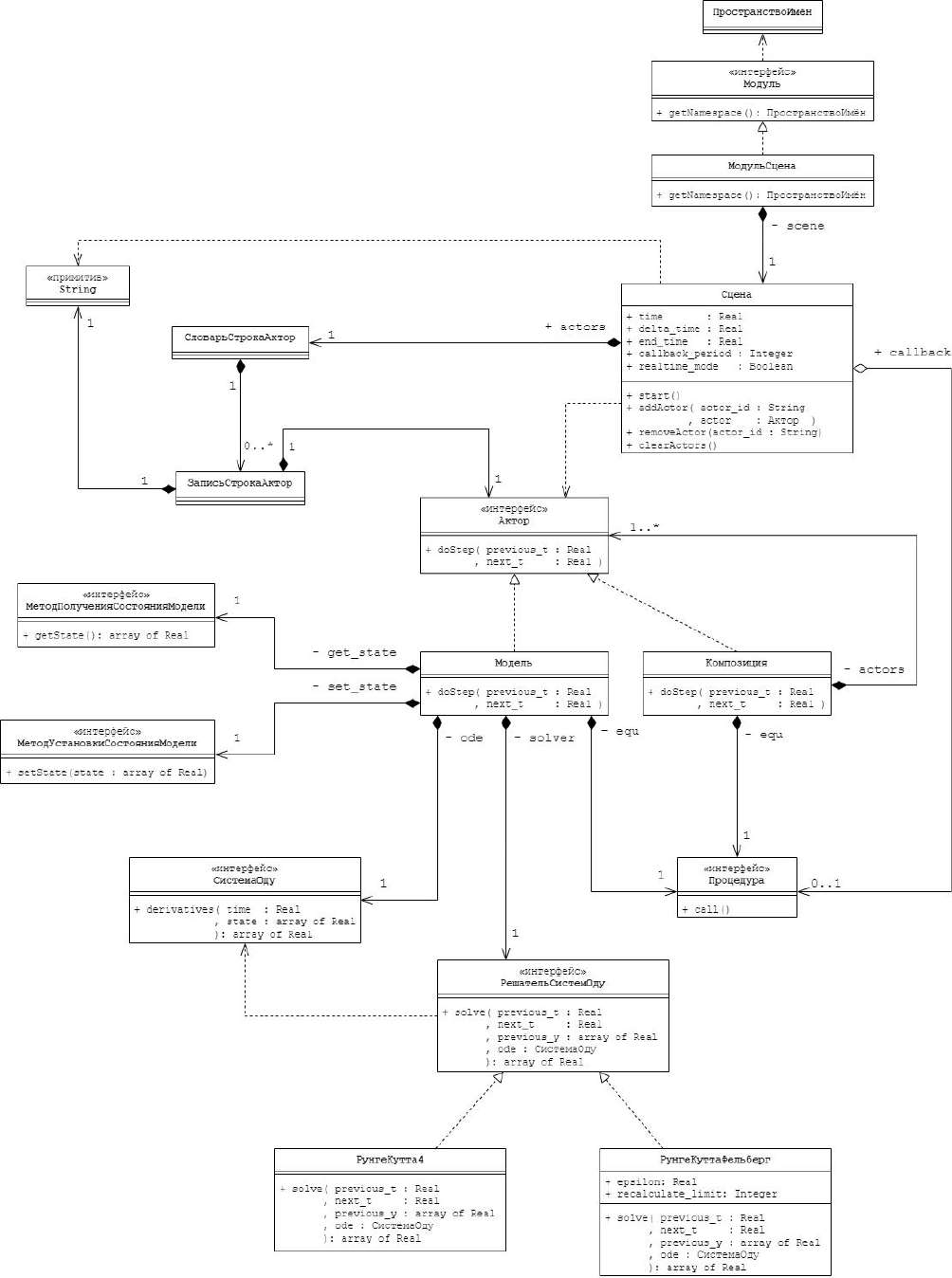


Рисунок 13 – Диаграммы классов модуля сцены

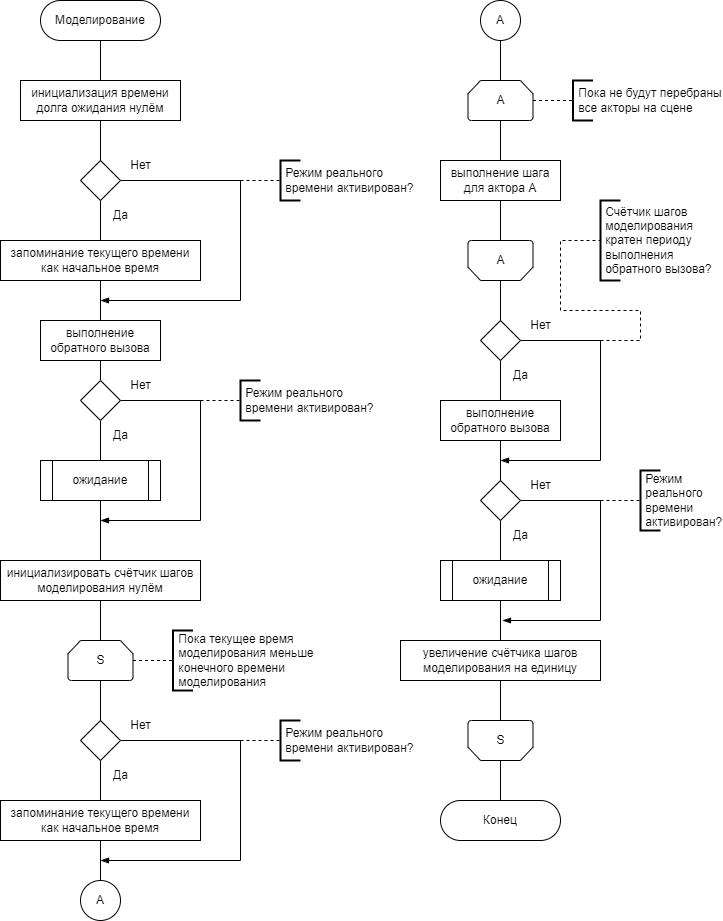


Рисунок 14 – Схемы алгоритмов модуля сцены

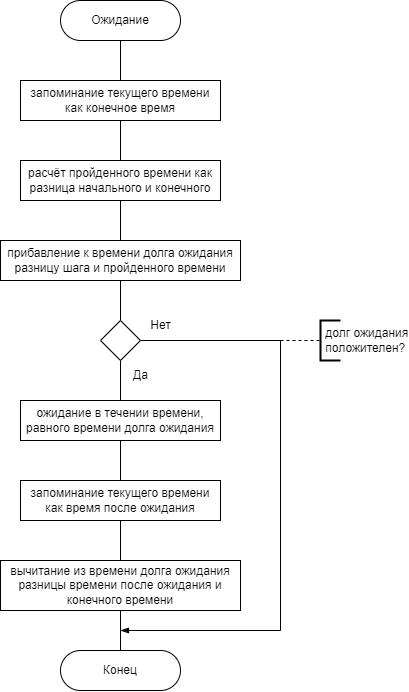


Рисунок 15 – Схемы алгоритмов модуля сцены

**3 Разработка технологии непрерывной доставки**

На кафедре ИУ-6 МГТУ им. Баумана ведётся разработка системы SIMODO – адаптивной системы моделирования, предназначенной для упрощения создания моделей.

Система SIMODO решает проблему построения сложных моделей, которые зачастую объединяют несколько предметных областей. Для решения этой проблемы предлагается использование предметно-ориентированных языков, каждый из которых описывает ту или иную предметную область и должен быть интуитивно понятен специалистам этой области. Система SIMODO предоставляет способ унификации поддержки предметно- ориентированных языков [17].

Практическая значимость системы SIMODO заключается в возможности эффективно описывать и просчитывать сложные модели.

Система SIMODO предназначена для платформ Windows и Linux и не обладает автоматизированной системой развёртывания.

Развёртывание ПО – это все действия, которые делают программную систему готовой к использованию. Данный процесс является частью жизненного цикла программного обеспечения.

Целью данной эксплуатационной практики является создание и развитие системы развёртывания для системы SIMODO.

Задачами ставятся:

1. Исследование видов установочных пакетов приложений.
2. Выбор установочного пакета для системы SIMODO.
3. Разработка системы формирования установочного пакета.
4. Автоматизация формирования установочного пакета.

**3.1 Исследование видов установочных пакетов**

Целевыми платформами системы SIMODO являются Windows и Linux. Дистрибутив Linux существует великое множество. Решено было остановится на одном из самых распространённых дистрибутиве Ubuntu версии Jammy [18], использующий установочные пакеты DEB и дистрибутиве отечественной разработки Alt Linux версии P10, использующий установочные пакеты RPM. Первый критерий к установочным пакетам – поддержка целевых платформ Ubuntu Jammy и Alt Linux P10.

Исходные тексты программ системы SIMODO расположены в репозитории системы DevOps Gitlab [19]. Для непрерывного развёртывания в данной системе используются виртуальные машины на основе Linux дистрибутивов, что определяет второй критерий для установочного пакета системы SIMODO – возможность сборки пакета на дистрибутиве Linux, в частности под целевую платформу Windows.

Если поставлять динамические библиотеки, необходимые для работы системы SIMODO, вместе с системой SIMODO, то размер установочного пакета будет значительно больше, чем установочный пакет, к которому операционная система сможет самостоятельно предоставить необходимые зависимости. Третьим и необязательным критерием выбора установочного пакета является наличие системы разрешения зависимостей.

Были выбраны и исследованы установочные пакеты RPM, DEB, NSIS, MSI, QtIFW. В таблице 2 представлены результаты исследования.

Таблица 2. Сравнение установочных пакетов

|  |  |  |  |
| --- | --- | --- | --- |
| **Установоч- ный пакет** | **Целевые платформы** | **Возможность сборки под Windows в Linux дистрибутиве** | **Наличие системы разрешения зависимостей** |
| RPM | Linux дистрибутивы с пакетным менеджером rpm | – | + |
| DEB | Linux дистрибутивы с пакетным менеджером dpkg | – | + |

Таблица 2. Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| **Установоч- ный пакет** | **Целевые платформы** | **Возможность сборки под Windows в Linux дистрибутиве** | **Наличие системы разрешения зависимостей** |
| NSIS | Windows | + | – |
| MSI | Windows | – | + |
| QtIFW | Windows, Linux | + | – |

Из таблицы 2 видно, что установочный пакет QtIFW наиболее полно соответствует выставленным критериям.

**3.2 Разработка автоматизированной системы развёртывания**

Выбранный установочный пакет QtIFW собирается с помощью одноимённого фреймворка, который прежде, чем собирать установочный пакет, необходимо собрать из исходников [20]. Сам фреймворк требует аналогичной процедуры для статической библиотеки Qt. Для дистрибутивов Linux, в которых нет возможности установить утилиту linuxdeployqt, так же приходится собирать из исходников. Для сборки фреймворков под платформу Windows используется среда кросскомпиляции mxe, которую было решено заранее скомпилировать из исходников и разместить в репозитории Gitlab[20].

В приложении Б приведён листинг скрипта YAML, который используется для сборки установочного пакета в процессе непрерывного развёртывания Gitlab.

Чтобы собрать установочный пакет системы SIMODO, необходимо интегрировать данный процесс в проект системы SIMODO. В приложении А представлен листинг скрипта интеграции сборки установочного пакета в проект графического редактора системы SIMODO. С помощью переменных окружения контролируется вариант сборки под Windows или Linux.

Реализованная автоматизированная система развёртывания системы SIMODO формирует установочные пакеты системы SIMODO как артефакты конвейера системы непрерывного развёртывания Gitlab[20].

Получаемые установочные пакеты позволяют автономно устанавливать систему SIMODO под целевые платформы.

**ЗАКЛЮЧЕНИЕ**

родлрлдордлордлрдлордлорд

Ордлордлордлрдлордлордлор

родлрлдордлордлрдлордлорд

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. SIMODO в репозитории МГТУ им. Н.Э. Баумана. [Электронный ресурс]. URL: https://bmstu.codes/lsx/simodo (дата обращения: 07.07.2022).
2. Иванова Г.С., Жильцов А.И., Фетисов М.В., Чулин Н.А., Юдин А.Е. Адаптивная система моделирования. — Автоматизация. Современные технологии, номер 11 за 2020 год, стр. 500.
3. Официальная страница LSP [Электронный ресурс]. — URL: https://microsoft.github.io/language-server-protocol/ (дата обращения: 30.11.2022)
4. Популярные библиотеки C++ [Электронный ресурс]. — URL: https://en.cppreference.com/w/cpp/links/libs (дата обращения: 30.11.2022)
5. Библиотека Boost [Электронный ресурс]. — URL: https://www.boost.org/ (дата обращения: 30.11.2022)
6. Документация на фреймворк eCal [Электронный ресурс]. — URL: https://continental.github.io/ecal/\_api/ecal\_root.html (дата обращения: 30.11.2022)
7. Фреймворк ACE [Электронный ресурс]. — URL: http://www.dre.vanderbilt.edu/~schmidt/ACE.html (дата обращения: 30.11.2022)
8. Библиотека POCO [Электронный ресурс]. — URL: https://pocoproject.org/ (дата обращения: 30.11.2022)
9. Документация на фреймворк Qt [Электронный ресурс]. — URL: https://doc.qt.io/qt-5/ (дата обращения: 30.11.2022)
10. Межпроцессное взаимодействие в Qt [Электронный ресурс]. — URL: https://doc.qt.io/qt-6/ipc.html (дата обращения: 30.11.2022) 11. Неименованный канал [Электронный ресурс]. — URL: https://www.baeldung.com/linux/anonymous-named-pipes (дата обращения: 30.11.2022)
11. Десять популярных языков программирования для настольных приложений в 2021 году. [Электронный ресурс]. URL: https://www.decipherzone.com/blog-detail/top-programming-languages-for-desktop- apps-in-2021 (дата обращения: 07.02.2023)
12. SIMODO в репозитории МГТУ им. Н.Э. Баумана. [Электронный ресурс]. URL: https://bmstu.codes/lsx/simodo (дата обращения: 07.02.2023).
13. Ещё раз про семь основных методологий разработки / Хабр [Электронный ресурс]. URL: https://habr.com/ru/company/edison/blog/269789 (дата обращения: 07.02.2023)
14. Иванова, Г.С. Технология программирования: Учебник для вузов – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. 238 с.
15. IBM Developer [Электронный ресурс]. URL: https://developer.ibm.com (дата обращения: 07.02.2023).
16. ГОСТ 19.701-90 «ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения»
17. Иванова Г.С., Жильцов А.И., Фетисов М.В., ЧулинН.А., ЮдинА.Е. Адаптивная система моделирования. – Автоматизация. Современныетехнологии, номер 11 за 2020 год, стр. 500.
18. Рейтинг дистрибутивов Linux. [Электронный ресурс]. URL: https://www.tecmint.com/top-most-popular-linux-distributions (дата обращения: 07.07.2022).
19. SIMODO в репозитории МГТУ им. Н.Э. Баумана. [Электронныйресурс]. URL: https://bmstu.codes/lsx/simodo (дата обращения: 08.07.2022).
20. Документация QtIFW. [Электронный ресурс]. URL: https://doc.qt.io/qtinstallerframework/ifw-getting-started.html (дата обращения: 05.07.2022).
21. Документация Gitlab. [Электронный ресурс]. URL: https://doc.qt.io/qtinstallerframework/ifw-getting-started.html (дата обращения: 10.07.2022).