МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«**Вятский государственный университет**»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Отчет
Комплекс лабораторных работ по дисциплине
«Предметно-ориентированные автоматизированные информационные системы»

Выполнил студент группы ИВТ-42 _____/Рзаев А. Э./
Проверил преподаватель кафедры ЭВМ _____/Караваева О. В./

Киров 2019

1 Описание предметной области

В данном комплексе лабораторных работ в качестве исследуемой системы рассматривается программная модель диспетчера памяти.

2 Описание предметной области

В настоящее время в области образования все больше производится автоматизация контроля знаний учащихся и освоения нового материала.

Одним из способов автоматизации являются специальные программные модели, эмулирующие работу какой-либо системы. С их помощью студенты имеют возможность достаточно подробно изучить ее работу, принципы и особенности. В совокупности с теоретическим материалом это позволяет увеличить степень освоения новых знаний по данной дисциплине и повысить качество обучения в целом.

Такие программные модели достаточно широко используются при выполнении лабораторных работ по дисциплине «Операционные системы». К сожалению, качество некоторых приложений оставляет желать лучшего, что затрудняет изучение нового материала. Поэтому было принято решение выполнить анализ и доработку наиболее проблемной модели – диспетчера памяти операционной системы.

2.1 Функции планировщика оперативной памяти

Функциями планировщика оперативной памяти в многозадачных ОС являются:

- отслеживание свободной и занятой памяти;
- первоначальное и динамическое выделение памяти процессам приложений и самой операционной системе и освобождение памяти по завершении процессов;
- настройка адресов программы на конкретную область физической памяти;
- полное или частичное вытеснение кодов и данных процессов из оперативной памяти (ОП) на диск, когда размеры ОП недостаточны для размещения всех процессов, и возвращение их в ОП;
- защита памяти, выделенной процессу, от возможных вмешательств со стороны других процессов;
- дефрагментация памяти.

Для изучения функций планировщика была разработана текущая программная модель. Студентам предлагается выполнить задание, состоящее из последовательности заявок, которые необходимо обработать, выступив в роли планировщика.

2.2 Модель памяти

В данной модели адресное пространство разбито на 256 страниц памяти по 4096 байт каждая. Наименьшая единица адресного пространства, доступная для выделения процессу – страница. Выделять можно только целое число страниц.

Процессам в данной модели могут присваиваться идентификаторы (PID) от 0 до 255 включительно.

Непрерывная область памяти, состоящая из одной и более страниц и имеющая начальный адрес и размер, называется блоком памяти.

Каждый блок памяти имеет следующие параметры:

- адрес начала блока;
- размер блока в страницах;
- идентификатор процесса, которому принадлежит данный блок.

Память процессам при каждом запросе выделяется в виде одного блока памяти.

Состояние памяти определяется совокупностью всех блоков памяти. Каждая страница адресного пространства должна находиться в одном и только одном блоке памяти, т. е. последовательность блоков памяти, упорядоченных по начальному адресу, должна полностью покрыть адресное пространство.

Под начальным состоянием памяти подразумевается такое состояние памяти, при котором все адресное пространство покрыто одним свободным блоком памяти с начальным адресом 0 и размером 256.

2.3 Интерфейс пользователя

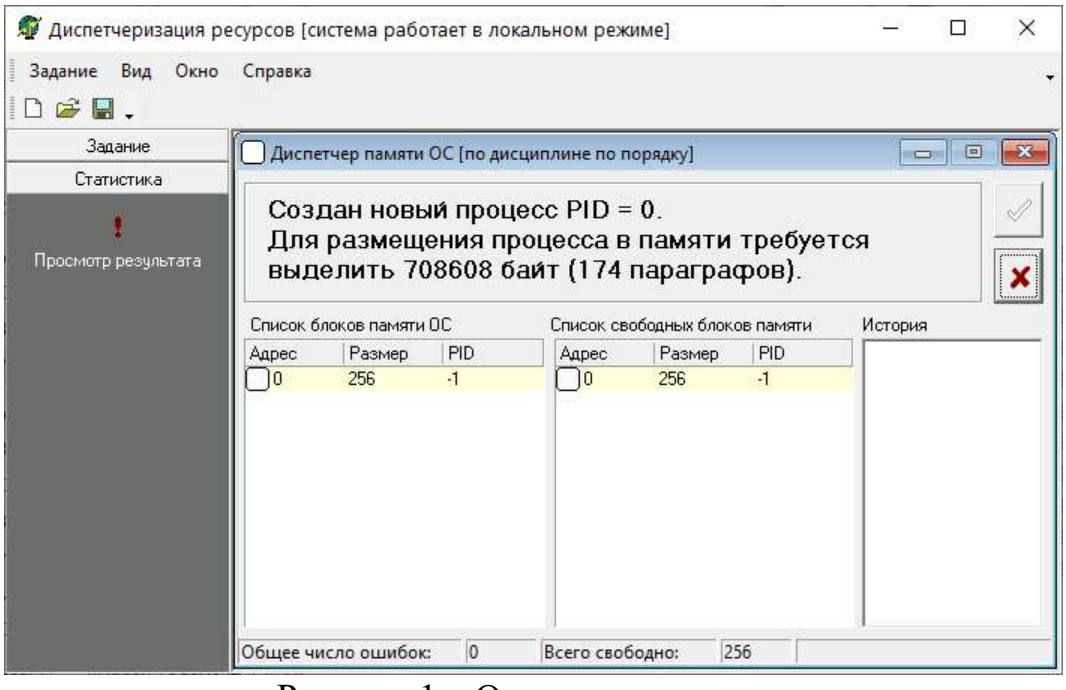

Рисунок 1 – Основное окно программы

Главное окно программы разделено на следующие области:

- заголовок с названием дисциплины;
- описание текущей заявки;
- список блоков памяти ОС;
- список свободных блоков памяти;
- блок кнопок: «Подтвердить» и «Отклонить»;
- список обработанных пользователем заявок;
- строка статуса.

В верхней части окна присутствует описание заявки и количество выполненных заявок. Задача пользователя – корректно обработать поступающие заявки.

В данной программной модели имеются следующие типы заявок:

- создание нового процесса;
- завершение процесса;
- выделение памяти существующему процессу;
- освобождение памяти, которой владеет процесс.

В левой части окна находится список блоков памяти. Для каждого блока указан его начальный адрес, размер в страницах, а также идентификатор процесса, владеющего данным блоком («-1» – нет владельца).

Пользователь имеет возможность выполнить следующие действия:

- ответить на запрос отказом, сразу нажав на кнопку "Отклонить";
- выделить память процессу. Для этого необходимо щелкнуть правой кнопкой мыши на свободном блоке в списке блоков памяти ОС и выбрать пункт «Выделить приложению». После этого в появившемся окне нужно ввести идентификатор процесса, которому выделяется память, а также количество выделяемых страниц;
- освободить память. Для этого необходимо щелкнуть правой кнопкой мыши на занятом блоке в списке блоков памяти ОС и выбрать пункт «Освободить»;
- объединить два свободных блока в один. Для этого необходимо щелкнуть правой кнопкой мыши на свободном блоке в списке блоков памяти ОС и выбрать пункт «Объединить со следующим», Данное действие необходимо выполнять каждый раз, когда образуются соседние свободные блоки;
- уплотнить (дефрагментировать память). Для этого необходимо щелкнуть правой кнопкой мыши на списке блоков памяти ОС и выбрать пункт «Уплотнение памяти»;
- подтвердить действия, нажав на кнопку "Подтвердить".

Основные дисциплины выбора свободного блока памяти:

- выбор первого подходящего блока. В этом случае список свободных блоков сортируется по адресу и выбирается первый блок, размер которого больше или равен требуемому размеру;

- выбор самого подходящего блока. Список свободных блоков сортируется по размеру (от меньшего к большему) и выбирается первый подходящий по размеру блок;
- выбор самого неподходящего блока. Самый неподходящий блок – это блок с максимальным размером. Список свободных блоков необходимо отсортировать в порядке убывания размера и выбрать первый блок.

В правой части экрана находится список выполненных шагов, которые хранят историю действий пользователя. Элемент в списке помечается красным цветом, если действия пользователя на данном шаге были некорректны. При наведении указателя мыши на элемент списка отображается информация о событии и действиях пользователя на данном шаге. Для отмены произвольного числа шагов необходимо щелкнуть мышью на элемент списка, до которого необходимо очистить историю. Отменённые шаги помечаются серым цветом. До тех пор, пока не выполнено какое-либо действие в окне диспетчера, можно восстановить действия пользователя, щелкнув на записи, соответствующей отмененному шагу.

В нижней части окна расположена строка статуса, в которой отображается общее число ошибок, которые допустил пользователь в процессе выполнения лабораторной работы, а также количество свободных страниц оперативной памяти.

По достижении заданного числа шагов выводится сообщение о выполнении лабораторной работы, а также количество неисправленных ошибок.

2.4 Недостатки

В текущей программной модели были найдены следующие недостатки:

- приложение доступно только для ОС Windows. Пользователи других операционных систем должны запускать Windows в виртуальной машине либо использовать другие средства по запуску Windows-приложений в других ОС;
- нестабильность. В ходе выполнения лабораторной работы программная модель несколько раз аварийно завершалась, из-за чего результаты выполненной работы безвозвратно терялись;
- ошибки при проверке пользовательских действий. В некоторых случаях было замечено, что программная модель принимала правильную последовательность действий как ошибочную. Это в совокупности с нестабильностью программы усложняет изучение студентами программной модели и, как следствие, увеличивает время на выполнение лабораторной работы;
- нечеткость, «размытость» интерфейса на дисплеях со сверхвысоким разрешением (HiDPI);

– в связи с утерей исходного кода программы и сложностью дизассемблирования определить формат файла задания не представляется возможным, что препятствует разработке новых вариантов заданий.

Лабораторная работа по данной теме (управление памятью в ОС) имеет важную роль в закреплении лекционного материала и предоставляет студентам возможность изучить более подробно устройство ОС.

Из-за того, что исходный код программной модели утерян, исправить ошибки и недостатки в текущей модели не представляется возможным. Поэтому было принято решение разработать новую программную модель, повторяющую функционал текущей, в которой будут исправлены вышеописанные ошибки и недостатки.

3 Техническое задание

3.1 Наименование и область применения

Наименование программы – "Модель диспетчера памяти операционный системы".

Программа предназначена для закрепления студентами лекционного материала по дисциплине «Операционные системы», а именно: управление памятью в операционных системах.

3.2 Назначение разработки

Функциональным назначением программы является предоставление студентам возможности изучить работу диспетчера памяти во время выполнения лабораторных работ.

Программа должна эксплуатироваться на ПК студентов, преподавателей и на ПК, установленных в учебных аудиториях Вятского государственного университета. Особые требования к конечному пользователю не предъявляются.

3.3 Требования к программе

3.3.1 Требования к функциональным характеристикам

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

– функции генерации задания;
– функции сохранения текущего прогресса выполнения задания в файл;
– функции загрузки задания с сохраненным прогрессом выполнения из файла;

- функции подсчета количества ошибок, сделанных в ходе выполнения задания;
- функции просмотра и отмены действий, выполненных в ходе прохождения задания.

3.3.2 Входные данные программы

Входными данными системы является запрос пользователя на выполнение лабораторной работы.

3.3.3 Выходные данные программы

Выходными данными системы является отчет с результатами выполнения задания.

3.4 Требования к надёжности

3.4.1 Требования к обеспечению надежного функционирования системы

Надежное (устойчивое) функционирование системы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- организацией бесперебойного питания технических средств;
- использованием лицензионного программного обеспечения;

3.4.2 Время восстановления после отказа

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

3.5 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер, включающий в себя:

- x86-совместимый процессор с тактовой частотой не меньше 1.0 ГГц;
- дисплей с разрешением не меньше, чем 1024x768;
- не менее 1 гигабайта оперативной памяти;
- не менее 100 мегабайт свободного дискового пространства;
- клавиатура, мышь.

- Системные программные средства, используемые программой, должны быть представлены следующими операционными системами:
- 64-разрядная ОС Windows 7/8/8.1/10;
- 64-разрядная ОС Ubuntu 16.04 и выше.

3.6 Требования к программной документации

Состав программной документации должен включать в себя:

1) техническое задание;
2) программу и методики испытаний;
3) руководство пользователя;
4) техническую документацию;
5) исходный код.

3.7 Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1) разработка технического задания;
2) рабочее проектирование;
3) внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка программы;
2) разработка программной документации;
3) испытания программы.

На стадии внедрения должен быть выполнен этап подготовки и передачи программы Заказчику.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;
2) определение и уточнение требований к техническим средствам;
3) определение требований к программе;
4) определение стадий, этапов и сроков разработки программы и документации на неё;
5) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованием п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1) разработка, согласование, утверждение программы и методики испытаний;
2) проведение приемо-сдаточных испытаний;
3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена подготовка и передача программы и программной документации в эксплуатацию Заказчику.

4 Структурный подход

Для проведения анализа и реорганизации бизнес-процессов предназначено CASE-средство AllFusion ERwin Process Modeler, поддерживающее методологии: IDEF0, IDEF3, DFD.

Построение модели системы начинается с описания функционирования системы в целом.

Входными данными системы является запрос пользователя на выполнение лабораторной работы.

Выходными данными системы является отчет с результатами выполнения задания.

Для своей работы системы руководствуется алгоритмами обработки заявок.

Работу в системе осуществляет студент.

Контекстная диаграмма представлена на рисунке 2.

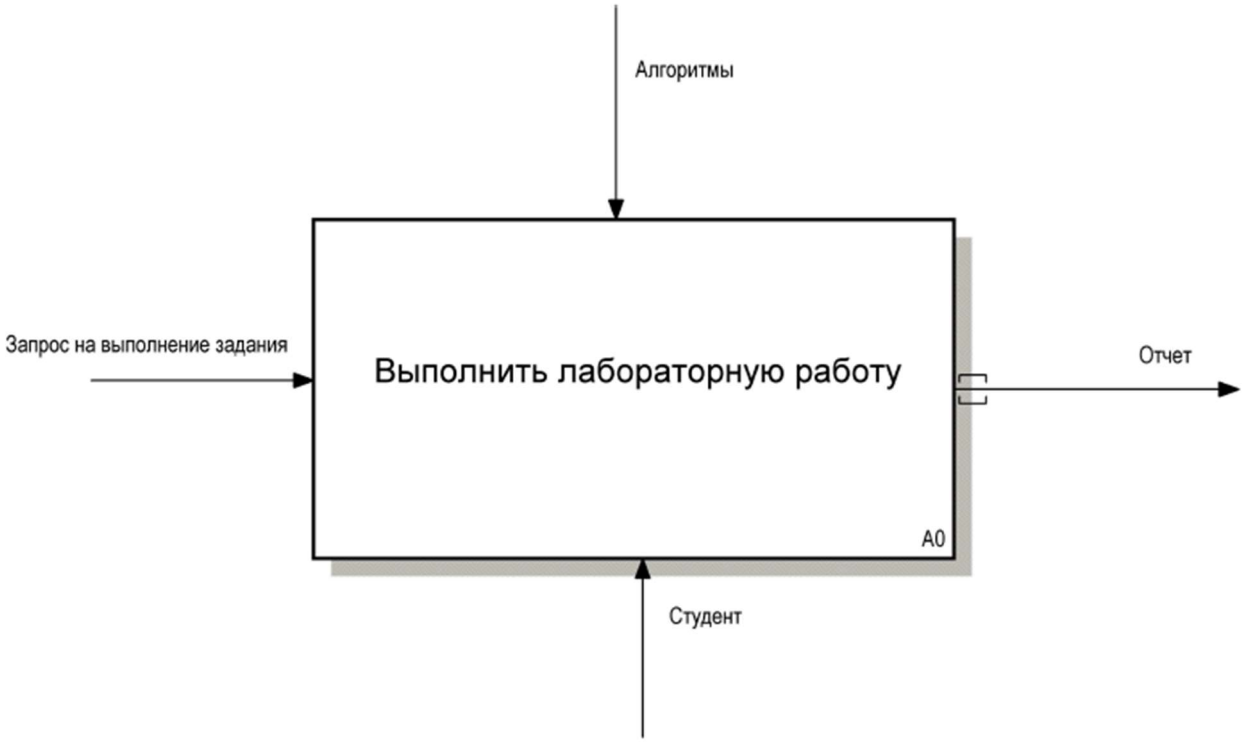

Рисунок 2 – Контекстная диаграмма IDEF0

Процесс выполнения лабораторной работы делится на 3 этапа: подготовка задания, выполнение задания и подготовка отчета. На этапе подготовки задания пользователь может либо сгенерировать новое задание, либо, если у него есть файл с заданием, загрузить его и продолжить выполнение задания. Этап выполнения задания является основным. Здесь пользователь должен выступить в качестве диспетчера памяти и обработать заявки, поступающие от процессов. По окончании выполнения задания программа запоминает количество сделанных ошибок пользователем и на следующем этапе подготавливает отчет о выполненной работе.

Диаграмма декомпозиции системы представлена на рисунке 3.

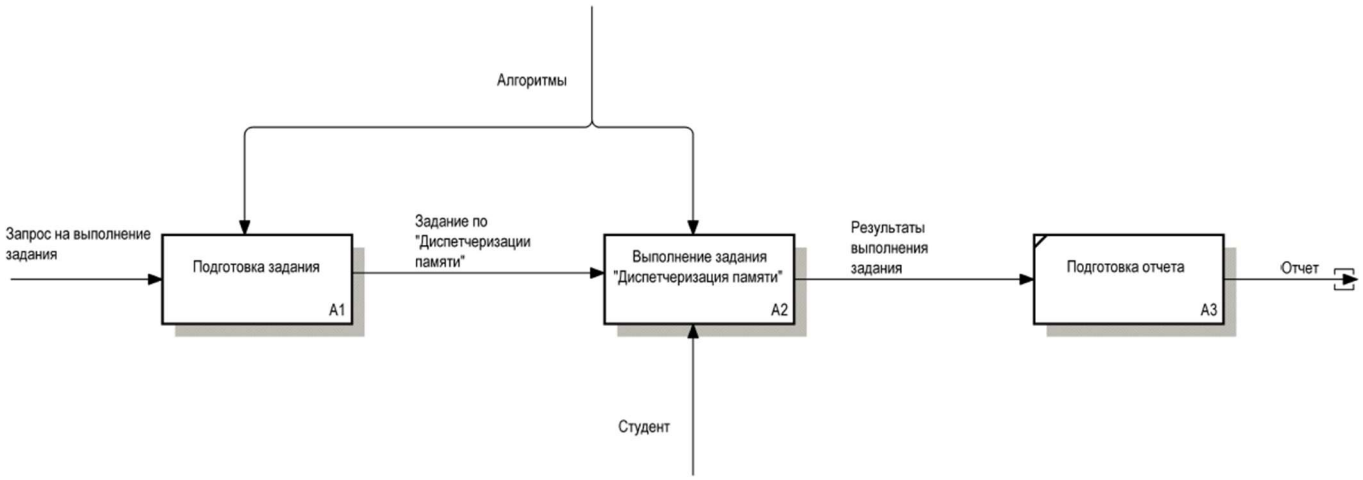

Рисунок 3 – Диаграмма декомпозиции системы

На этапе выполнения задания программа проверяет, правильно ли студент обработал очередную заявку и если ошибок нет, то переходит к следующей заявке. В противном случае выводится сообщение об ошибке. Сохранить ход выполнения задания можно перед проверкой выполненных студентом шагов. Для наглядного представления последовательности выполняемых процессов была построена модель в нотации IDEF3, которая представлена на рисунке 4.

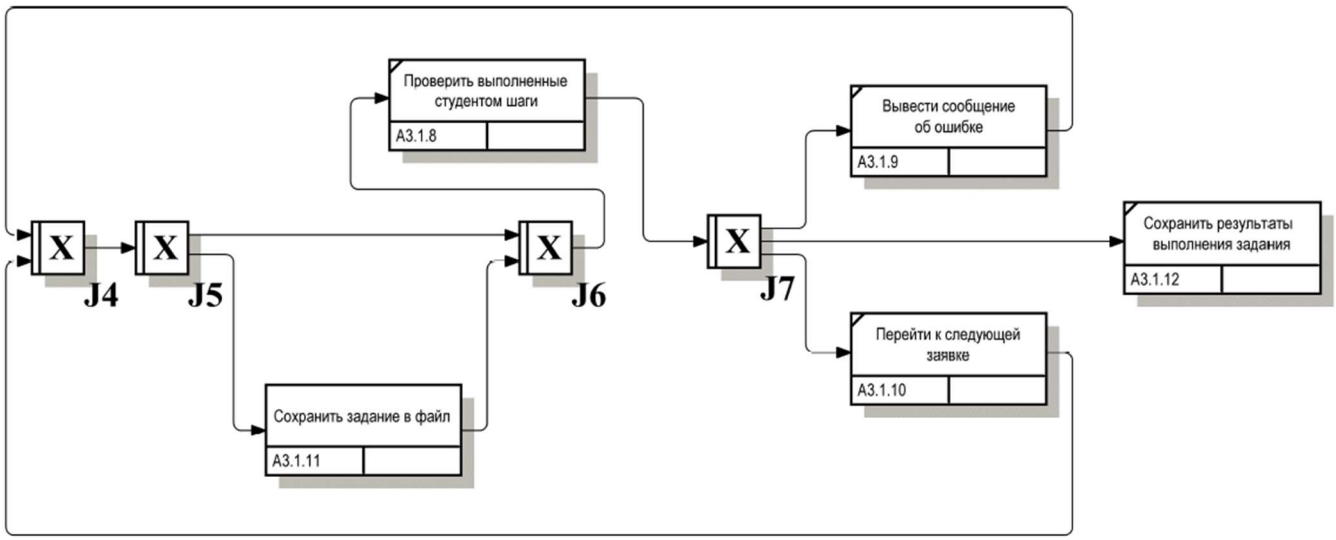

Рисунок 4 – Декомпозиция этапа выполнения задания

На этапе подготовки задания, как уже было сказано выше, студент может либо сгенерировать новое задание. Также он может через меню приложения открыть диалог выбора файла и загрузить из выбранного файла задание. Однако файл может быть поврежден. Для стабильной работы программной модели осуществляется проверка загруженного задания. В случае отсутствия ошибок заявки загружаются в программу, и студент приступает к выполнению задания. В противном случае предлагается либо сгенерировать задания, либо выбрать другой файл задания.

Последовательность описанных действий представлена в виде модели в нотации IDEF3 на рисунке 5.

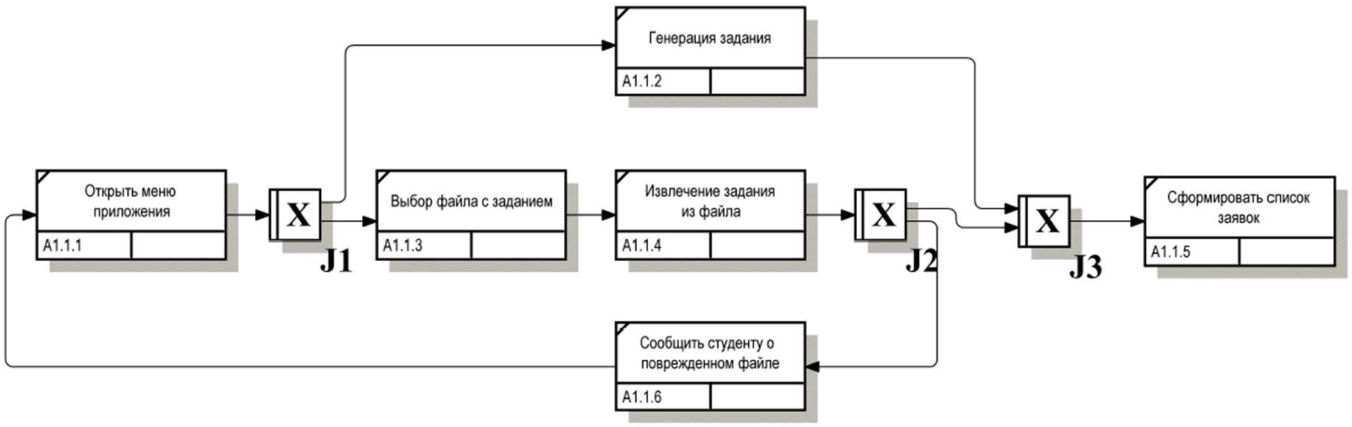

Рисунок 5 – Декомпозиция этапа подготовки задания

Извлечение и проверка задания выполняется в несколько этапов, а именно:

1) чтение содержимого файла. Если прочитать содержимое файла не удалось, то пользователю выводится соответствующее сообщение об ошибке;
2) содержимое файла (текст) передается парсеру JSON, который составляет JSON-объект задания. Если не удалось выполнить парсинг текста, то пользователю выводится соответствующее сообщение об ошибке;
3) на основе полученного JSON-объекта задания выполняется построение списка заявок, состояние памяти, определение дисциплины планировщика и извлечение статистики. На данном этапе могут возникнуть ошибки из-за некорректных типов заявок, дисциплины, состояния памяти или просто некорректных значений. О возникших ошибках сообщается пользователю через графический интерфейс;
4) на основе полученных данных выполняется проверка задания на корректность в случае, если в файл были внесены изменения, направленные на подмену результатов. При обнаружении таких изменений пользователю выводится сообщение об ошибке;
5) полученный объект задания передается в модуль графического интерфейса.

DFD-диаграмма данного процесса представлена на рисунке 6.

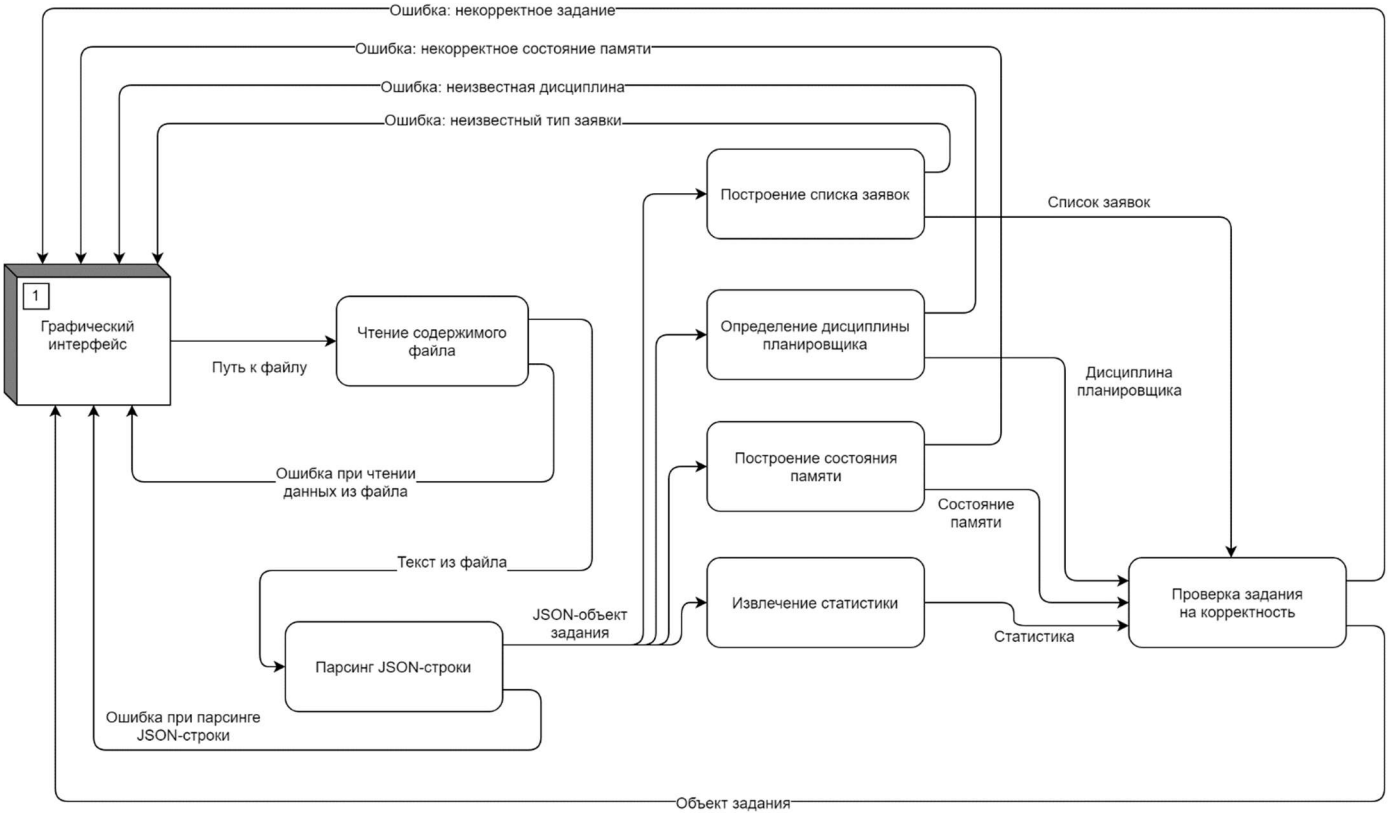

Рисунок 6 – Декомпозиция процесса извлечения задания из файла

Процесс сохранения задания в файл состоит также из нескольких шагов. Студент должен через меню приложения открыть диалог выбора файла, в нем выбрать нужную папку и имя файла для сохранения. Если такой файл уже есть, то пользователь должен либо подтвердить перезапись файла, либо выбрать другую папку. После того, как местоположение и имя файла выбрано, в него записывается задание. Последовательность описанных действий представлена в виде модели в нотации IDEF3 на рисунке 7.

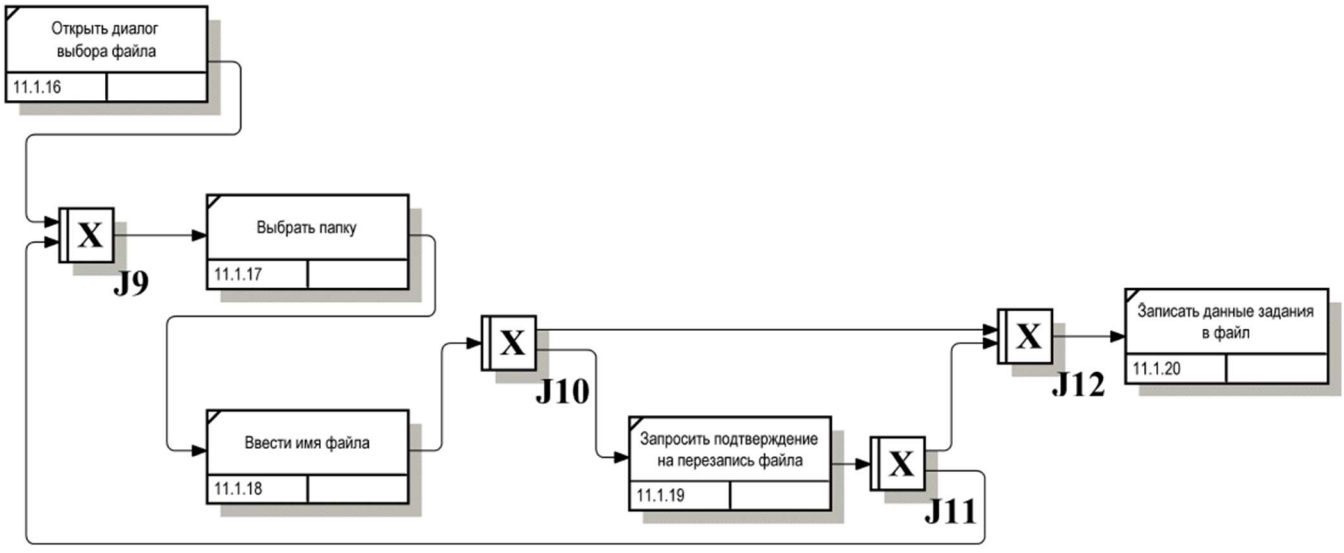

Рисунок 7 – Декомпозиция процесса сохранения задания в файл

5 Объектно-ориентированный подход

5.1 Диаграмма вариантов использования

Use Case или диаграмма вариантов использования отражает отношения между актерами и прецедентами. Суть данной диаграммы состоит в том, что проектируемая система представляется в виде множества актеров, которые взаимодействует с системой с помощью вариантов использования.

В качестве актера в проектируемой системе будет выступать Студент. Студент взаимодействует с двумя прецедентами: «Подготовить задание» и «Выполнить задание».

Вариант использования «Подготовить задание» содержит расширения «Сгенерировать задание» и «Загрузить из файла».

Прецедент «Выполнить задание» состоит из включений «Обработать заявку», «Проверить выполненные шаги» и расширения «Сохранить задание в файл».

Диаграмма вариантов использования представлена на рисунке 8.

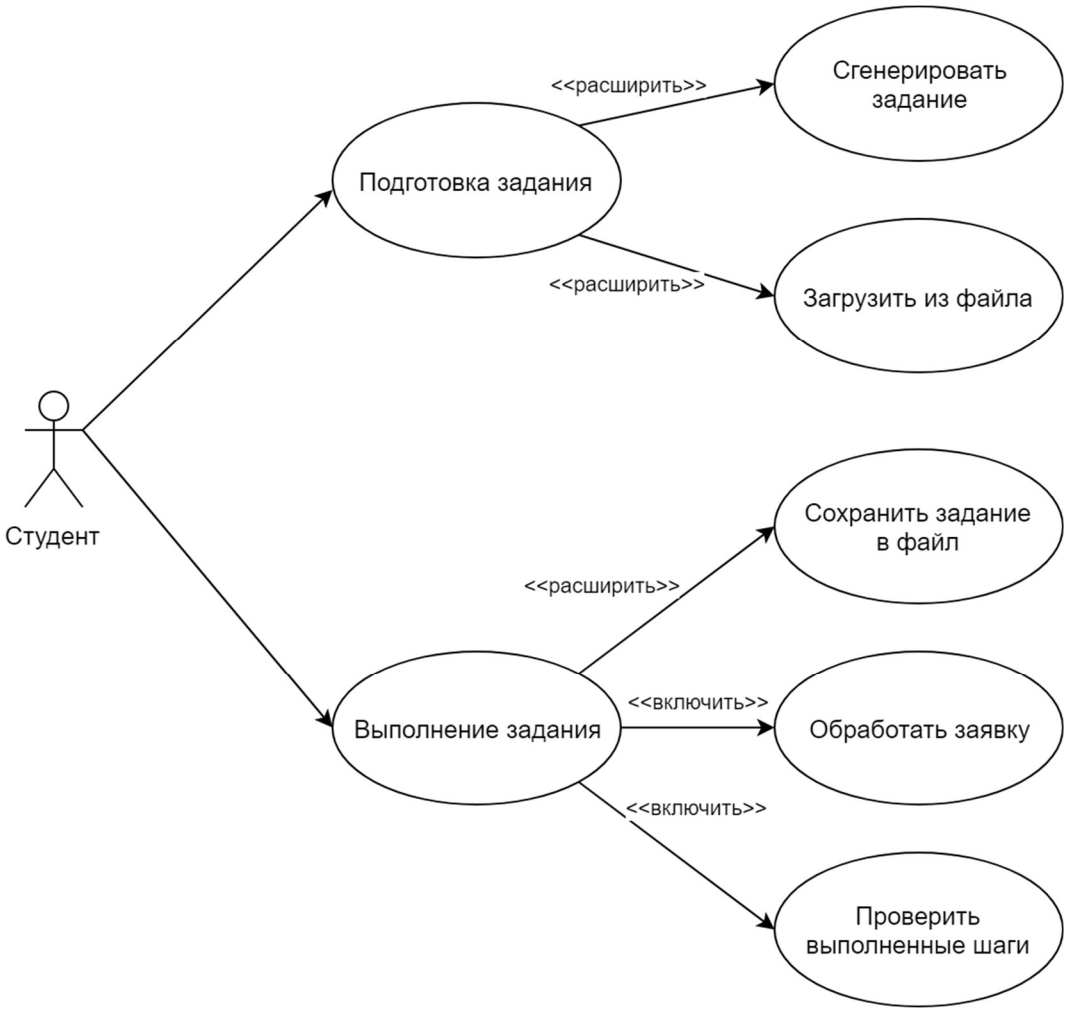

Рисунок 8 – Диаграмма вариантов использования

5.2 Диаграмма последовательности

Диаграмма последовательности предназначена для моделирования взаимодействия объектов проектируемой системы во времени и обмена сообщениями между ними в рамках одного прецедента (варианта использования). Основные понятия данной системы связаны с понятиями объект и сообщение. На диаграмме объекты в основном представляют экземпляры класса или сущности, обладающие поведением. Неотъемлемой частью объекта является его линия жизни. Она показывает, в течение какого периода существует объект.

Диаграмма последовательности строится для каждого варианта использования. Необходимо построить две диаграммы последовательности (для вариантов использования «Подготовка задания» и «Выполнение задания»).

Диаграммы последовательности представлены на рисунках 9 – 11.

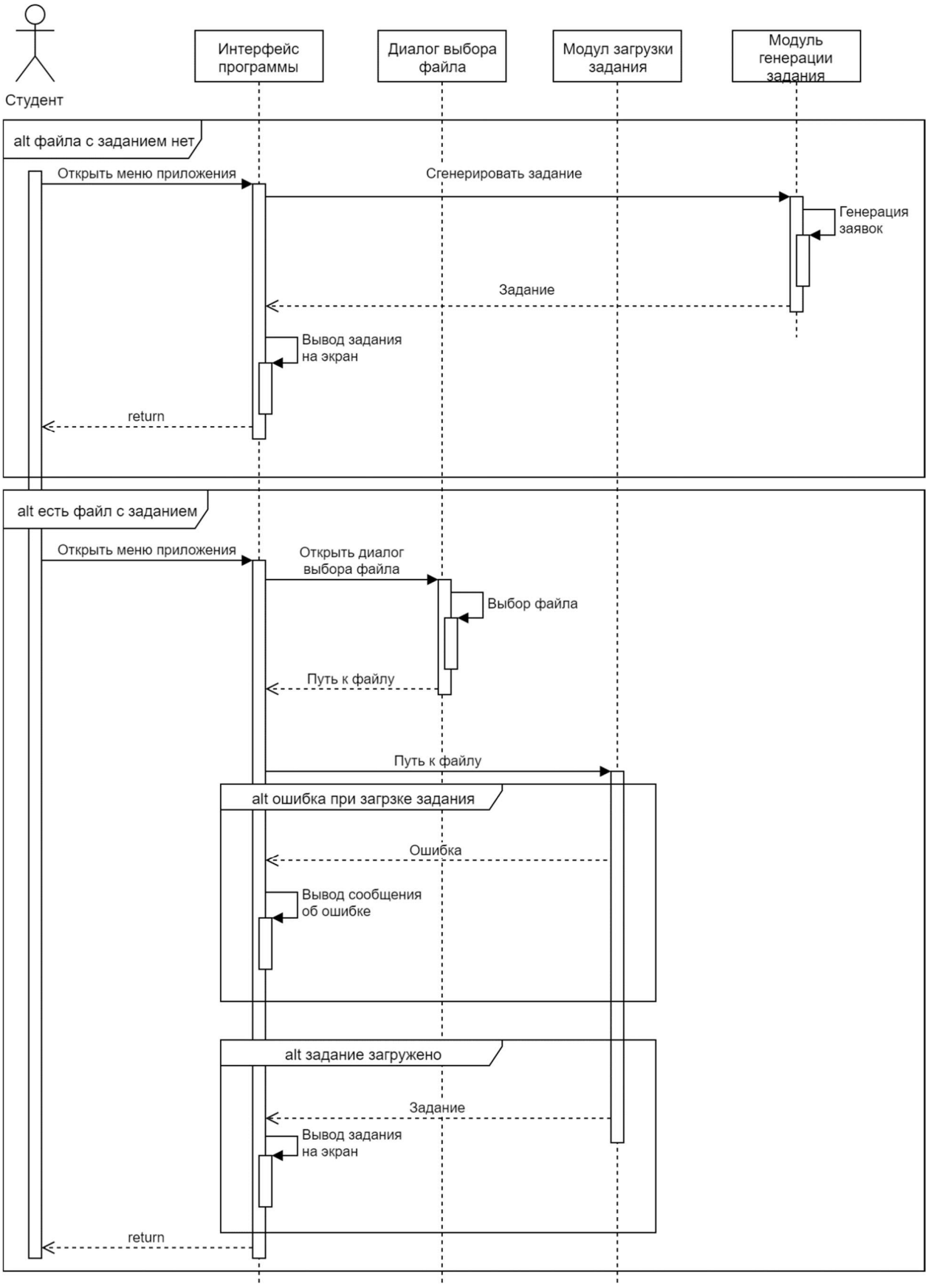

Рисунок 9 – Диаграмма последовательности для варианта использования «Подготовка задания»

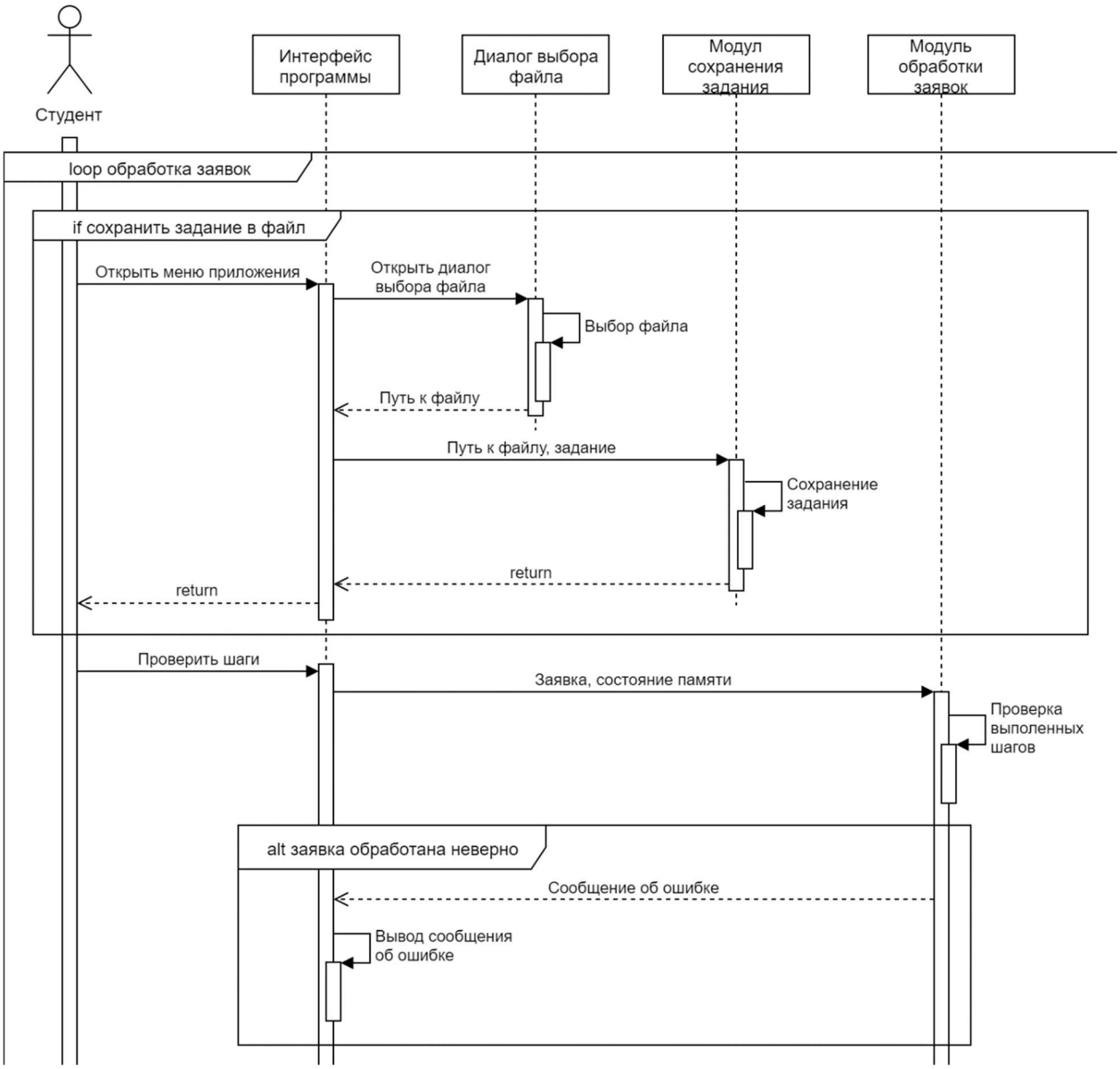

Рисунок 10 – Диаграмма последовательности для прецедента «Выполнение задания» (часть 1)

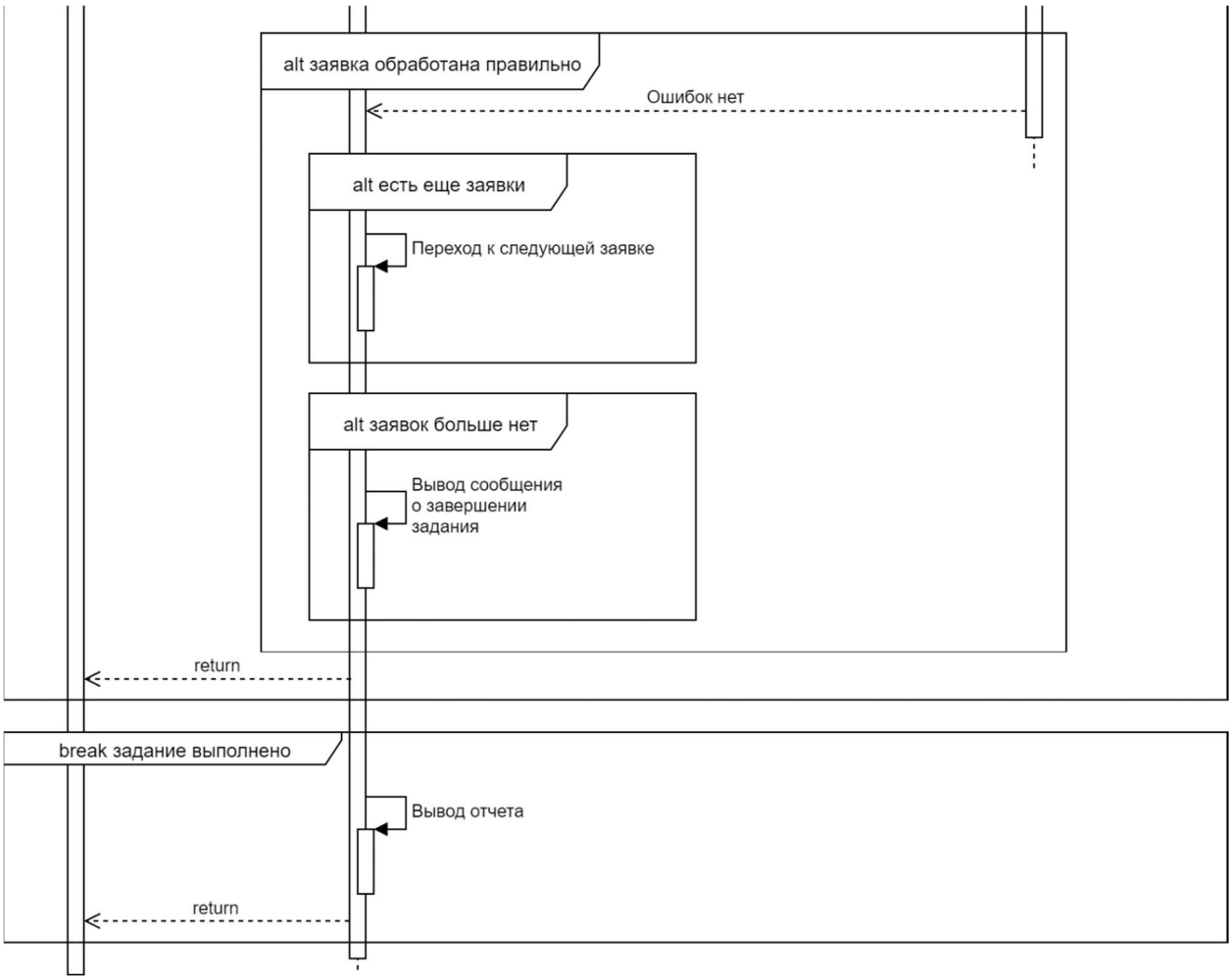

Рисунок 11 – Диаграмма последовательности для прецедента «Выполнение задания» (часть 2)

5.3 Диаграмма классов

Диаграмма классов используются при моделировании информационных систем наиболее часто. Она являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. На диаграммах классов показываются классы, интерфейсы и отношения между ними, но не отражает динамическое поведение объектов, изображенных на ней классов.

Диаграмма классов проектируемой системы состоит из следующих классов:

– «MainWindow» – класс основного окна графического интерфейса; определяет работу пользователя с разрабатываемой системой;

– «MemoryTaskWindow» – класс, содержащий элементы интерфейса, обеспечивающие отображение информации о текущем задании;

– «AboutWindow» – класс справочного окна программы;

– «MemoryTask» – класс, описывающий задание;

– «MemoryState» – класс, описывающий состояние памяти;

– «MemoryBlock» – класс, описывающий блок памяти;

– «CreateProcess», «TerminateProcess», «AllocateMemory», «FreeMemory» – классы заявок;

– «Request» – обобщенный класс для заявок различных типов (тип-сумма);

– «AbstractStrategy» – класс, реализующий обобщенные алгоритмы обработки заявок;

– «LeastAppropriateStrategy», «MostAppropriateStrategy», «FirstAppropriateStrategy» – классы, в которых содержатся специализированные для каждой дисциплины функции обработки заявок.

Диаграмма классов представлена на рисунке 12.

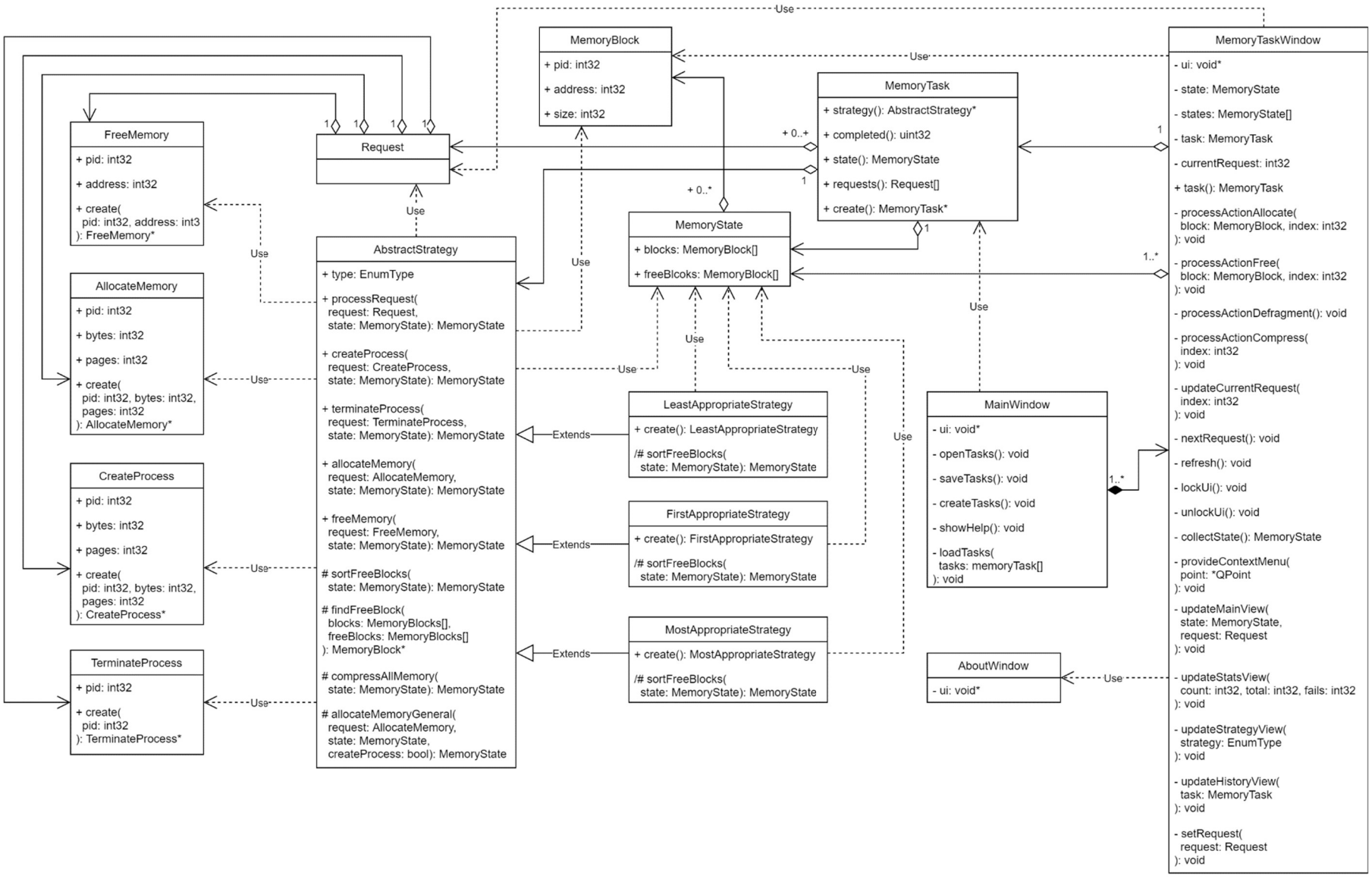

Рисунок 12 – Диаграмма классов

6 Выводы

В результате выполнения лабораторной работы были изучены объектно-ориентированный и структурный подходы проектирования информационных систем. Также разработаны различные диаграммы, упрощающие написание программного продукта, такие как: IDEF0, IDEF3, DFD, ER, UML.

Каждый из рассматриваемых подходов имеет свои преимущества и недостатки. В зависимости от решаемых задач более целесообразно использовать тот или иной подход.

К достоинствам структурного подхода можно отнести:

– возможность проведения глубокого анализа бизнес-процессов, выявления узких мест;

– применение диаграмм IDEF0, IDEF3 и DFD обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов.

Недостатком структурного подхода является быстрое увеличение глубины рассматриваемой системы при большой сложности конечной системы, что затрудняет дальнейшую работу со схемой.

К достоинствам объектно-ориентированного подхода можно отнести:

– сравнительная легкость, наглядность, эффективность моделей;

– возможность автоматической генерации кода на основе построенных моделей.

Недостатки объектно-ориентированного подхода:

– невозможность проведения детального анализа бизнес-процессов;

– неполнота и незавершенность некоторых видов диаграмм, возможность их неверной интерпретации.