Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Воронежский государственный лесотехнический университет

имени Г.Ф. Морозова»

Лабораторная работа № 6

По дисциплине: Тестирование ИС

На тему: «Разработка алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования»

Выполнил студент группы: ИС1-224-ОТ Севидов А.Р.

Проверил преподаватель СПО: Фёдоров В. Ю.

Воронеж 2025

Цель работы: разработка алгоритма для создания программного продукта и анализ предметной области

**Краткие теоретические сведения**

Необходимо написать программу, которая будет выполнять действия над матрицами: умножения, сложения, вычитания, транспонирования. Программа должна решать введенные вручную матрицу в форму. Для удобства пользователя программа должна иметь интуитивно понятный интерфейс.

Выбор методов и разработка основных алгоритмов решения

В программе используется следующий алгоритм работы: в программе есть формы, в которые вводятся элементы матриц, элементы переводятся из String типа в Integer. Затем нужно нажать кнопку соответствующего действия. Выполняется алгоритм решения матриц и результат выводится в элемент DataGridView.

Для построения блок-схем использовалась программа Microsoft Office Visio 2013. С её помощью можно составлять различные диаграммы и схемы, в том числе, блок-схемы.

Разработка кода программного продукта на основе готовой спецификации на уровне модуля

Калькулятор матриц реализован на языке программирования C# в среде программирования Microsoft Visual Studio Ultimate 2013. Выбор языка C# обусловлен тем, что он современный и популярный объектно-ориентированный язык программирования, а среда Microsoft Visual Studio Ultimate 2013 является мощным средством, позволяющим быстро создать программу, обладающую графическим оконным интерфейсом.

На форме располагается 3 элемента DataGridView, в них будут размещаться матрицы.

Так же 4 Button для выполнения действий над матрицами.

Использование инструментальных средств на этапе отладки программного модуля

При отладке программного продукта необходимо воспользоваться командой меню Отладка. В меню отладка существуют ряд команд, назначение которых представлено ниже.

Окна – открывает в интегрированной среде окно Точки останова, которое дает доступ ко всем точкам останова данного решения. Показывает в интегрированной среде окно Вывод.

Окно Вывод – это бегущий журнал множества сообщений, выдаваемых интегрированной средой, компилятором и отладчиком. Поэтому эта информация относится не только к сеансу отладки, а также открывает в интегрированной среде окно Интерпретация, которое позволяет выполнять команды:

начать отладку – запускает приложение в режиме отладки;

присоединиться к процессу – позволяет прикрепить отладчик к выполняющемуся процессу (исполняемому файлу). например, если запущено приложение без отладки, то можете потом прикрепиться к этому выполняющемуся процессу и начать отладку;

исключения – открывает диалоговое окно Исключения, которое позволяет выбрать способ останова отладчика для каждого исключительного состояния;

шаг с заходом – запускает приложение в режиме отладки. для большинства проектов выбор команды шаг с заходом означает вызов отладчика на первой выполняемой строке приложения. таким образом, можно войти в приложение с первой строки;

шаг с обходом – когда вы не находитесь в сеансе отладки, то команда шаг с обходом просто запускает приложение точно так же, как это сделала бы кнопка run;

точка останова – включает или выключает точку останова на текущей (активной) строке кода текстового редактора. эта опция неактивна, если в интегрированной среде нет активного кодового окна;

создавать точку останова – активирует диалоговое окно создавать точку останова позволяющее указать имя функции, для которой необходимо создать точку останова;

удалить все точки останова – удаляет все точки останова из текущего решения;

очистить все подсказки по данным – деактивирует (без удаления) все точки останова текущего решения;

параметры и настройки – Прерывать выполнение, когда исключения пересекают границу домена приложения или границу между управляемым и машинным кодом.

Проведение тестирования программного модуля по определенному сценарию Оценочное тестирование, которое также называют «тестированием системы в целом»

целью которого является тестирование программы на соответствие основным требованиям. Эта стадия тестирования особенно важна для программных продуктов. Включает следующие виды:

тестирование удобства использования – последовательная проверка соответствия программного продукта и документации на него основным положениям технического задания;

тестирование на предельных объемах – проверка работоспособности программы на максимально больших объемах данных, например, объемах текстов, таблиц, большом количестве файлов и т.п.;

тестирование на предельных нагрузках – проверка выполнения программы на возможность обработки большого объема данных, поступивших в течение короткого времени;

тестирование удобства эксплуатации – анализ психологических факторов, возникающих при работе с программным обеспечением; это тестирование позволяет определить, удобен ли интерфейс, не раздражает ли цветовое или звуковое сопровождение и т.п.;

тестирование защиты – проверка защиты, например, от несанкционированного доступа к информации;

тестирование производительности – определение пропускной способности при заданной конфигурации и нагрузке;

тестирование требований к памяти – определение реальных потребностей в оперативной и внешней памяти;

тестирование конфигурации оборудования – проверка работоспособности программного обеспечения на разном оборудовании;

тестирование совместимости – проверка преемственности версий: в тех случаях, если очередная версия системы меняет форматы данных, она должна предусматривать специальные конвекторы, обеспечивающие возможность работы с файлами, созданными предыдущей версией системы;

тестирование удобства установки – проверка удобства установки;

тестирование надежности – проверка надежности с использованием математических моделей;

тестирование восстановления – проверка восстановления программного обеспечения, например, системы, включающей базу данных, после сбоев оборудования и программы;

тестирование удобства обслуживания – проверка средств обслуживания, включенных в программное обеспечение;

тестирование документации – тщательная проверка документации, например, если документация содержит примеры, то их все необходимо попробовать;

тестирование процедуры – проверка ручных процессов, предполагаемых в системе. Естественно, целью всех этих проверок является поиск несоответствий техническому заданию. Считают, что только после выполнения всех видов тестирования программный продукт может быть представлен пользователю или к реализации. Однако на практике обычно выполняют не все виды оценочного тестирования, так как это очень дорого и трудоемко. Как правило, для каждого типа программного обеспечения выполняют те виды тестирования, которые являются для него наиболее важными. Так базы данных обязательно тестируют на предельных объемах, а системы реального времени – на предельных нагрузках.

Оформление документации на программное средство

Созданный программный продукт предназначен для выполнения арифметических действий над матрицами.

Чтобы запустить программу нужно запустить приложение.

Для того чтобы создать матрицы, необходимо ввести размерности матрицы и нажать кнопки «Построить». Затем ввести данные в матрицу и выбрать желаемое действие.

Программа имеет удобный интерфейс и предоставляет возможность с легкостью решать матрицы произвольных размерностей.

Моделирование бизнес- процессов предметной области

Основные элементы и понятия IDEF0. Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия:

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде прямоугольника и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, “производить услуги”, а не “производство услуг”).

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом:

Верхняя сторона имеет значение "Управление" (Control);

Левая сторона имеет значение "Вход" (Input);

Правая сторона имеет значение "Выход" (Output);

Нижняя сторона имеет значение "Механизм" (Mechanism).

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер.

Вторым элементом методологии IDEF0 является понятие интерфейсной дуги (Arrow). Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое уникальное наименование (Arrow Label). По требованию стандарта, наименование должно быть оборотом существительного.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название “входящей”, “исходящей” или “управляющей”. Кроме того, “источником” (началом) и “приемником” (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом “источником” может быть

только выходная сторона блока, а “приемником” любая из трех оставшихся.

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь по крайней мере одну управляющую интерфейсную дугу и одну исходящую. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

При построении IDEF0 – диаграмм важно правильно отделять входящие интерфейсные дуги от управляющих, что часто бывает непросто.

В реальном процессе рабочему, производящему обработку, выдают заготовку и технологические указания по обработке (или правила техники безопасности при работе со станком). Ошибочно может показаться, что и заготовка, и документ с технологическими указаниями являются входящими объектами, однако это не так. На самом деле в этом процессе заготовка обрабатывается по правилам, отраженным в технологических указаниях, которые должны соответственно изображаться управляющей интерфейсной дугой.

Например, в случае рассмотрения предприятий и организаций существуют пять основных видов объектов: материальные потоки (детали, товары, сырье и т.д.), финансовые потоки (наличные и безналичные, инвестиции и т.д.), потоки документов (коммерческие, финансовые и организационные документы), потоки информации (информация, данные о

намерениях, устные распоряжения и т.д.) и ресурсы (сотрудники, станки, машины и т.д.). При этом в различных случаях входящими и исходящими интерфейсными дугами могут отображаться все виды объектов, управляющими только относящиеся к потокам документов и информации, а дугами-механизмами только ресурсы.

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Третьим основным понятием стандарта IDEF0 является декомпозиция (Decomposition). Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции.

Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и обозначается идентификатором “А-0”.

В пояснительном тексте к контекстной диаграмме должна быть указана цель (Purpose) построения диаграммы в виде краткого описания и зафиксирована точка зрения (Viewpoint).

Определение и формализация цели разработки IDEF0 – модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь. Например, если мы моделируем деятельность предприятия с целью построения в дальнейшем на базе этой модели информационной системы, то эта модель будет существенно отличаться от той, которую бы мы разрабатывали для того же самого предприятия, но уже с целью оптимизации логистических цепочек.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Например, функциональные модели одного и того же предприятия с точек зрения главного технолога и финансового директора будут существенно различаться по направленности их детализации. Это связано с тем, что в конечном итоге, финансового директора не интересуют аспекты обработки сырья на производственных станках, а главному технологу ни к чему прорисованные схемы финансовых потоков. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки,

Часто бывают случаи, когда отдельные интерфейсные дуги не имеет смысла продолжать рассматривать в дочерних диаграммах ниже какого-то определенного уровня в иерархии, или наоборот - отдельные дуги не имеют практического смысла выше какого-то уровня. Например, интерфейсную дугу, изображающую "деталь" на входе в функциональный блок "Обработать на токарном станке", не имеет смысла отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. С другой стороны, случается необходимость избавиться от отдельных "концептуальных" интерфейсных дуг и не детализировать их глубже некоторого уровня. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования. Обозначение "туннеля" (Arrow Tunnel) в виде двух круглых скобок вокруг начала интерфейсной дуги обозначает, что эта дуга не была. Последним из понятий IDEF0 является глоссарий (Glossary). Для каждого из элементов IDEF0: диаграмм, функциональных блоков, интерфейсных дуг существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. Например, для управляющей интерфейсной дуги “распоряжение об оплате” глоссарий может содержать перечень полей соответствующего дуге документа, необходимый набор виз и т.д. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Принципы ограничения сложности IDEF0-диаграмм. Обычно IDEF0-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в соответствующем стандарте приняты соответствующие ограничения сложности:

Ограничение количества функциональных блоков на диаграмме тремя-шестью. Верхний предел (шесть) заставляет разработчика использовать иерархии при описании сложных предметов, а нижний предел (три) гарантирует, что на соответствующей диаграмме достаточно деталей, чтобы оправдать ее создание;

Ограничение количества подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг четырьмя.

**Практическая часть**

Создал контекстную диаграммуА-0.

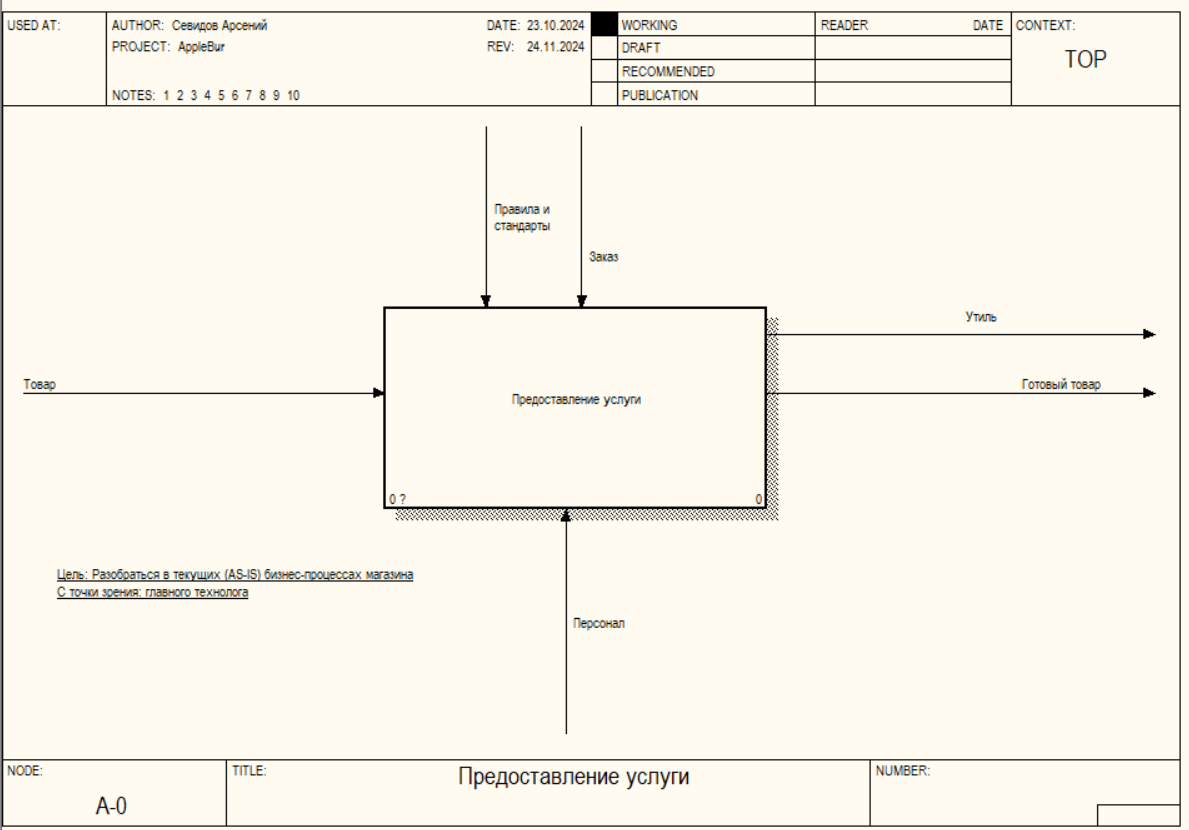


Рисунок 1 – Контекстная диаграмма

Таблица 1.Определение и типы стрелок на диаграмме.

|  |  |  |
| --- | --- | --- |
| Имя стрелки (Arrow Name) | Определение стрелки (Arrow Definition) | Тип стрелки (Arrow Type) |
| Запросы пользователей | Пользователь запрашивает доступ к определенному заказу | Управление (Control) |
| База данных техники | Файловые техники | Вход (Input) |
| Программа | Обеспечивает выполнение задач, связанных с готовым товаром | Механизм (Mechanism) |
| Имя стрелки (Arrow Name) | Определение стрелки (Arrow Definition) | Тип стрелки (Arrow Type) |
| Настройки пользователя | Система учитывает настройки пользователя | Управление (Control) |
| Наличие товара | Система проверяет наличие того или иного товара | Управление (Control) |
| Пользователь | Взаимодействие пользователя с процессом | Механизм (Mechanism) |
| Предоставление услуги | Отображает выбранный товар пользователем | Выход (Output) |

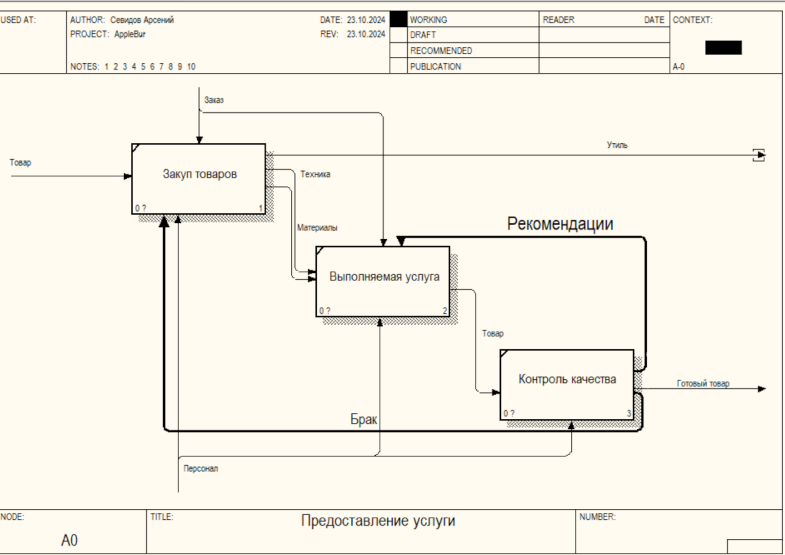


Рисунок 2 - Результат туннелирования стрелки на диаграмме А0.

Таблица 2. определения и статус для каждой работы

|  |  |  |
| --- | --- | --- |
| **Имя работы(Activity Name)** | **Определение(Definition)** | **Статус(Status)** |
| Обработка услуги пользователя | Обработка заданной услуги пользователем | working |
| Выполнение услуги | Сборка товара по выбранным данным | working |
| Контроль качества | Проверка соответсвия запроса с запросом пользователя | working |

Таблица 3. Внутренние стрелки декомпозиции

|  |  |  |
| --- | --- | --- |
| **Имя стрелки(ArrowName)** | **Определение стрелки(Arrow Definition)** | **Тип стрелки(ArrowType)** |
| Выбранный товар | Выполняет функцию передачи товара из обработки запроса | Обратная связь по входу (output-inputfeedback) |
| Название товара | Выполняет функцию передачи название товара из обработки запроса | Обратная связь по входу (output-inputfeedback) |
| Готовая услуга | Передаёт услугу в блок для проверки соответствий | Обратная связь по входу (output-inputfeedback) |
| Отображение товара | Отображает выбранный товар пользователем | Выход(Output) |
| Отсутствие товара | показывает, что отсутствует необходимый товара | Выход(Output) |
| Несоответствие услуг | Отображает, что услуги пользователя не соответствуют ожидаемым параметрам | выход-механизм(output-mechanism) |

Таблица4. Работы диаграммы декомпозиции А2

|  |  |
| --- | --- |
| **Имя работы(ActivityName)** | **Определение работы(ActivityDefinition)** |
| Получение данных | Отвечает за сбор, обработку и передачу информации из внешних источников в систему или приложение |
| Проверка доступности | Отвечает за мониторинг и оценку состояния ресурсов или сервисов, чтобы убедиться, что они доступны для использования |
| Формирование запроса | Обрабатывает входные данные, определяет параметры запроса и формирует его в требуемом формате для последующего выполнения |
| Активация запроса | Он инициирует процесс запроса, отправляет его на выполнение и обрабатывает ответ, получая данные или подтверждая выполнение операции |

Таблица 5.Стрелки диаграммы декомпозиции А2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя стрелки(ArrowName)** | **Источникстрелки(ArrowSource)** | **Типисточникастрелки(ArrowSourceType)** | **Назначениестрелки(ArrowDest.)** | **Типназначениястрелки(ArrowDest.Type)** |
| Выбранный товар | Границадиаграммы | Вход(Input) | Получение данных | Вход(Input) |
| Название товара | Границадиаграммы | Вход(Input) | Получение данных | Вход(Input) |

Продолжение таблицы 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Настройки пользователя | Граница  диаграммы | Контроль(Control) | Получение данных | Контроль(Control) |
| Валидация данных | Граница  диаграммы | Контроль(Control) | Получение данных | Контроль  (Control) |
| Проверка доступности |
| Доступность товара | Граница  диаграммы | Контроль  (Control) | Проверка доступности | Контроль  (Control) |
| Условия фильтрации | Граница  диаграммы | Контроль  (Control) | Формирование запроса | Контроль  (Control) |
| Шаблоны и стандарты формирования услуги | Граница  диаграммы | Контроль  (Control) | Формирование запроса | Контроль  (Control) |
| Полученные данные | Получение данных | Выход(Output) | Проверка доступности | Вход(Input) |
| Доступные элементы товара | Проверка доступности | Выход(Output) | Формирование запроса | Вход(Input) |
| Сформированная услуга | Формирование запроса | Выход(Output) | Активация запроса | Вход(Input) |

Продолжение таблицы 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СУБД | Границадиаграммы | Механизм(Mechanism) | Получение данных | Механизм(Mechanism) |
| Проверка доступности |
| Формирование услуги |
| Активация услуги |
| Готовая услуга | Выполнение услуги | Выход  (Output) | Вывод услуги | Выход  (Output) |

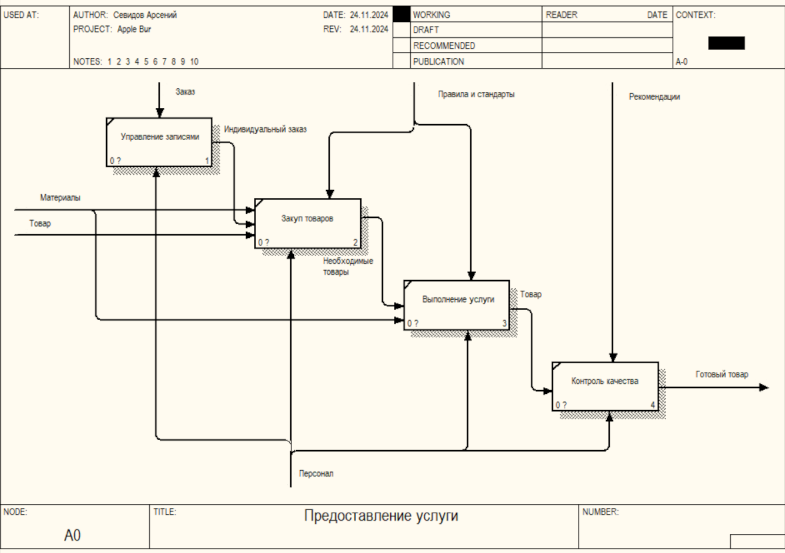


Рисунок 3 – Результаты диаграммы декомпозиции А2

Вывод: Я разработал алгоритмы для создания программного продукта и анализ предметной области.

**Контрольные вопросы**

1. **Каковы стадии жизненного цикла информационных систем,их основное содержание?**

* Формирование требований на основе анализа предметной области;
* Проектирование;
* Реализация;
* Внедрение (ввод системы в эксплуатацию);
* Эксплуатация (сопровождение проекта);

1. **Как представляется функциональная модель деятельности вместо дологии IDEF0?**

представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня.

1. **Каковы основные объекты диаграмм функциональной модели помето дологииIDEF0?**

Основными элементами диаграммы в нотации IDEF0 являются:

• блоки, в виде которых представлены процессы, функции, операции, действия (в зависимости от степени детализации);

• стрелки, в виде которых на диаграмме отражают информационные и материальные ресурсы, связанные с функциями.

1. **Что обозначают работы в диаграммах функциональной модели, как они отображаются по методологии IDEF0?**

Работы обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Работы изображаются в виде прямоугольников. Все работы должны быть названы и определены.

1. **Для чего предназначены стрелки в диаграммах функциональной модели, каковы их типы и виды?**

Стрелками изображаются входы и выходы бизнес-процесса на диаграмме. Они представляют материальные и нематериальные объекты, которые являются или необходимыми ресурсами для работы процесса, или результатами его выполнения.

1. **Для чего предназначен словарь стрелок?**

"Словарь стрелок" - это справочник уникальных названий стрелок моделей. Каждая именованная стрелка на диаграмме SADT неразрывно связана с одноименным объектом из справочника "Словарь стрелок".

1. **Каковы типы связей работ по методологииIDEF0?**

В IDEF0 различают 5 типов связей работ:

1.Связь по входу (output-input), когда стрелка выхода вышестоящей работы направляется на вход нижестоящей.

2.Связь по управлению (output-control), когда выход вышестоящей работы направляется на управление нижестоящей. Связь по входу показывает доминирование вышестоящей работы.

3.Обратная связь по входу (output-input feedback), когда выход нижестоящей работы направляется на вход вышестоящей.

4.Обратная связь по управлению (output-control feedback), когда выход нижестоящей работы направляется на управление вышестоящей.

5.Связь выход-механизм (output-mechanism), когда выход одной работы направляется на механизм другой.

1. **Что такое туннелирование стрелок, для чего оно нужно, каковы виды туннелирования?**

Туннелирование может быть применено для изображения малозначимых стрелок. Если на какой-либо диаграмме нижнего уровня необходимо изобразить малозначимые данные или объекты, которые не обрабатываются или не используются работами на текущем уровне, то их необходимо направить на вышестоящий уровень (на родительскую диаграмму). Если эти данные не используются на родительской диаграмме, их нужно направить ещё выше и т.д.