Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ ВСТРОЕННЫЕ СИСТЕМЫ

«Проектирование встроенной системы реального времени»

Проверила:
Авксентьева Е. Ю. _____
«____» _____ 2020 г.

Оценка _____

Выполнил:
Студент группы P3455
Федюкович С. А. _____
Вариант 15

Санкт-Петербург
2020

1 Оглавление

2 Техническое задание на разработку системы

Требуется разработать проект архитектуры программного обеспечения встроенной системы реального времени. В данной работе будет разработана система управления сканера.

3 Анализ требований к системе

К системе сканера выделить следующие внешние объекты:

1. Потенциальный пользователь
2. Пользователь
3. Обслуживающий персонал

Потенциальный пользователь формирует команду ”Начать сканирование”, в ответ на которую получает сообщение "Можно начать сканирование". Кроме того, потенциальный пользователь наблюдает за процессом сканирования.

Пользователь формирует команду ”Сканировать”, в ответ на которую (через некоторое время) получает сообщение "Можно начать сканирование". Кроме того, пользователь наблюдает за наблюдает за процессом сканирования и может послать сигнал ”Тревога”.

Обслуживающий персонал наблюдает за состоянием сканера и принимает сигналы тревоги в случае их появления. Контекстная диаграмма взаимодействия системы сканера с внешними объектами представлена на рис. 3.1.

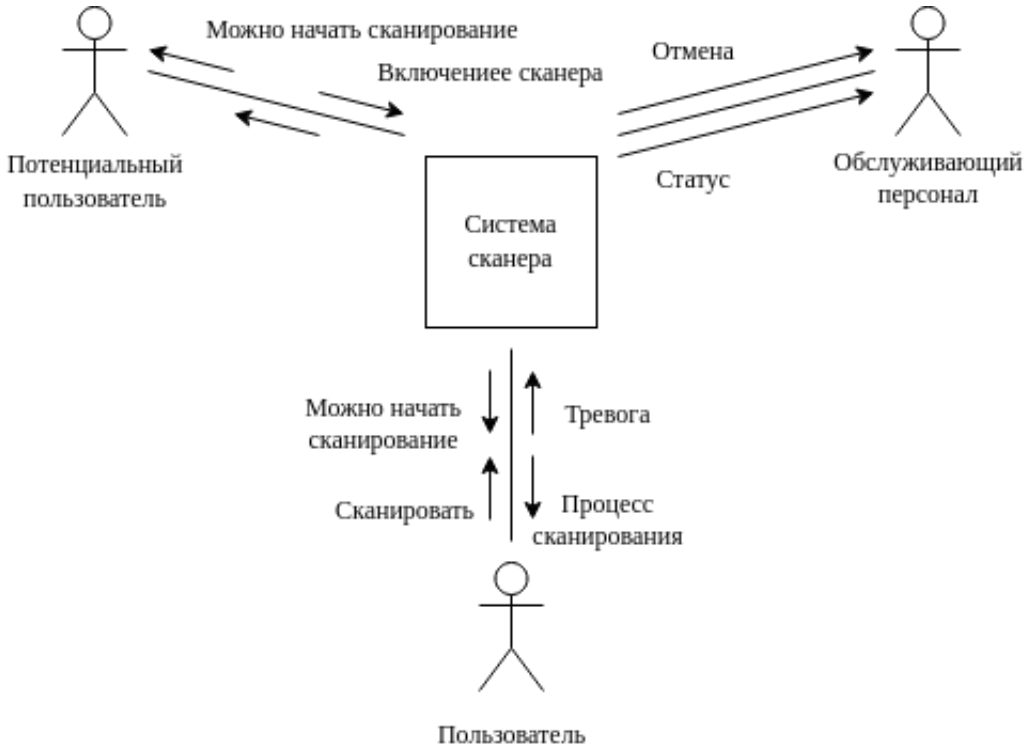

Рис. 3.1: Контекстная диаграмма взаимодействия системы сканера с внешними объектами

Сообщения являются базовым средством коммуникаций между объектами. На данном этапе определяются наиболее существенные свойства сообщений, к которым относятся следующие свойства:

- Содержание данных
- Тип синхронизации (с ожиданием и без ожидания)
- Периодичность

Наиболее важными являются сообщения, переносящие информацию о событиях. Спецификация должна включать перечисление событий и описание отклика на каждое событие.

Кроме того, для СРВ важны временные требования. Поэтому в таблицу описания событий в той или иной форме должны быть включены временные характеристики событий.

Если события периодические, то их временные характеристики специфицируются периодом. Если они эпизодические, то требуется специфицировать средний период и минимальный интервал.

Таблица 3.1 показывает список внешних событий системы сканера.

№	Событие	Отклик	Направление	Тип синхронизации	Периодичность	Временные требования
1	Потенциальный пользователь включает сканер	Если сканер свободен, то он включается. Если сканер занят, то запрос игнорируется	В систему	Без ожидания	Непериодическое	1 с.
2	Пользователь запрашивает скан	Подготовка к сканированию	В систему	Без ожидания	Непериодическое	1 с.
3	Пользователь устанавливает Кнопку «Пуск/Стоп» в положение «Стоп»	Сканер отменяет сканирование	В систему	Без ожидания	Непериодическое	1 с.
4	Пользователь устанавливает Кнопку «Пуск/Стоп» в положение «Пуск»	Сканер продолжает отрабатывать запросы	В систему	Без ожидания	Непериодическое	1 с.
5	Пользователь препятствует закрытию сканера	Сканер повторно открывается и запускается таймер закрытия крышки	В систему	Без ожидания	Непериодическое	1 с.
6	Истекло время нахождения сканера в состоянии «Открыта»	Начинается закрытие сканера	В систему	Без ожидания	Непериодическое	1 с.
7	Пользователь нажимает кнопку «Тревога»	Оповещение обслуживающего персонала	В систему	Без ожидания	Непериодическое	0.5 с.

Таблица 3.1: Список внешних событий системы сканера. Продолжение на следующей странице

8	Завершение сканирования	Открытие крышки	В систему	Без ожидания	Непериодическое	1 с.
9	Начало сканирования	Ожидание завершения сканирования	В систему	Без ожидания	Непериодическое	1 с.
10	Отключение питания	Включение механизма экстренного выключения	Из системы	Без ожидания	Непериодическое	1 с.
11	Отключение механизма блокировки	Начало отработки запросов сканера, если список запросов не пуст	В системы	Без ожидания	Непериодическое	1 с

Таблица 3.1: Продолжение таблицы

Поскольку отдельное внешние событие практически всегда порождает целый поток обмена сообщениями между объектами системы и её окружения, то после формирования списка внешних событий необходимо более детально рассмотреть отклик системы на каждое событие. Этой цели служат варианты использования системы.

Диаграммы вариантов использования представляют собой одно из средств описания реакции системы на определённые внешние события и являются детализацией контекстных диаграмм, рассмотренных выше.

Для выявления вариантов использования следует пытаться отвечать на вопросы типа:

1. Каковы цели создания системы?
2. Каковы наиболее важные функции системы?
3. Каковы дополнительные функции системы?
4. Как функционировала аналогичная система предыдущего поколения?

Анализ ответов на подобные вопросы позволяет определить:

1. Роли внешних объектов и системы при выполнении системой определенной функции.
2. Последовательности появления событий и сообщений при выполнении системой определённой функции.

На рис. 3.2 представлена диаграмма вариантов использования системы сканера.

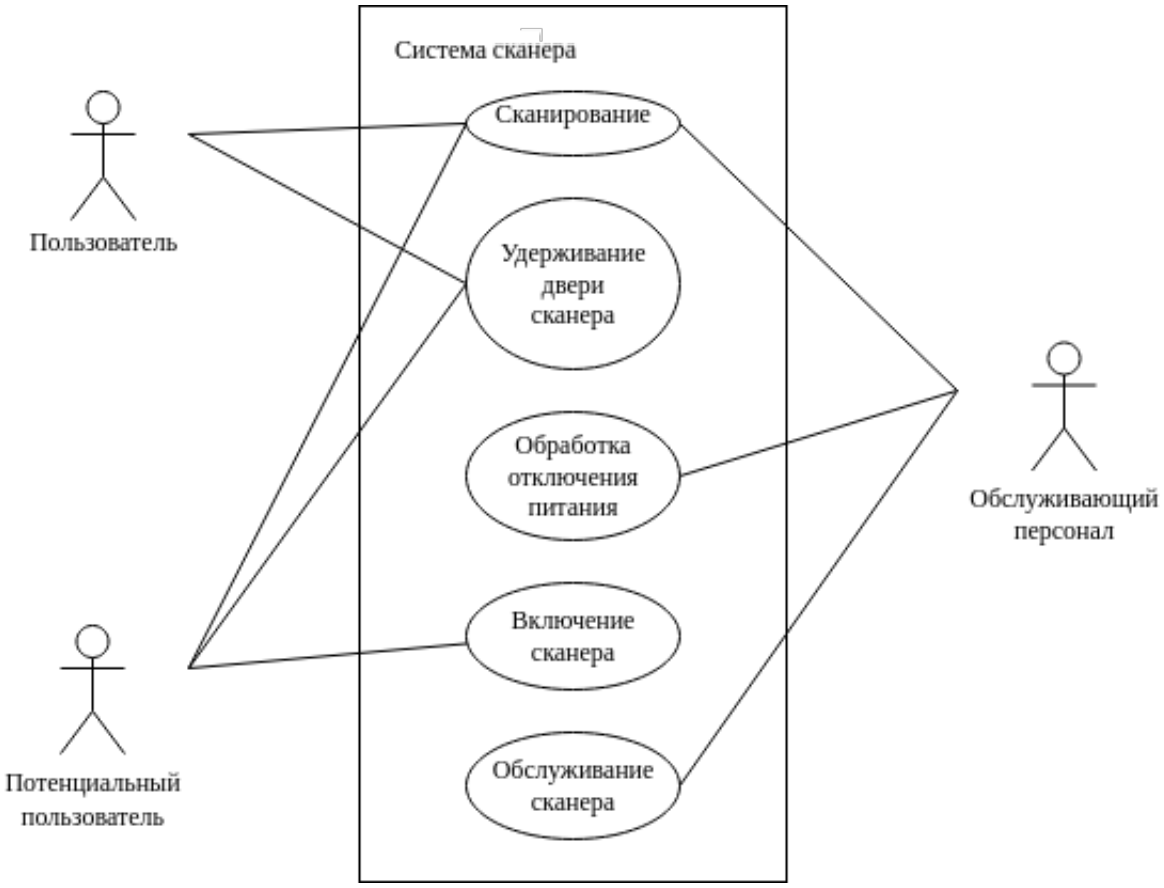

Рис. 3.2: Диаграмма вариантов использования системы сканера

Представленная диаграмма перечисляет варианты использования системы и показывает объекты, участвующие в каждом из вариантов.

Дальнейшей детализацией вариантов использования являются сценарии. Любой отдельный вариант использования порождает множество сценариев. Например, вариант «Включение сканера» может породить следующие сценарии:

1. Сканер уже включён
2. Сканер включается
3. Сканер имеет запрос, который должен быть обработан

В начальном этапе анализа в качестве объектов, участвующих в сценариях, выступают система и внешние объекты, определённые в контексте и вариантах использования. При последующем анализе система разбивается на объекты, и процесс построения сценариев может быть продолжен.

Сценарии обеспечивают инструмент для уточнения требований к системе с учётом разнообразия возможный ситуаций, в которых может оказаться система в процессе своего функционирования.

Построение и анализ сценариев — это сложная творческая деятельность.

В таблице 3.2 показан пример одного из возможных сценариев вызова лифта.

Шаг	Событие	Действие
	[Сканер своден и включён]	
1	Запрос сканера начать сканирование	Сканер начинает готовиться к сканированию и открывается
2	[Сканер готовится и открывается]	
3	Запрос сканера начать новое сканирования	Сканер запоминает запрос
4	Сканер подготовился к сканированию и открылся	
5	Пользователь 1 помещает объект для сканирования в сканер	Сканер закрывается и начинает сканировать объект
6	Сканирование заканчивается и сканер открывается	
7	Пользователь 1 забирает результаты и уходит	
8	Время таймера закрытия сканера истекает	Сканер закрывается и готовится к новому сканированию
9	Сканер готов к сканированию и открывается	
10	Пользователь 2 помещает объект для сканирования в сканер	Сканер закрывается и запускает процесс сканирования
11	Сканер заканчивает сканирование и открывается	Пользователь 2 забирает результаты и уходит
12	Сканер закрывается и остаётся свободным	

Таблица 3.2: Пример сценария вызова сканера

Две описания сценариев используют два варианта диаграмм — последовательные диаграммы и диаграммы сотрудничества.

Последовательные диаграммы описывают сценарии как последовательности передаваемых и принимаемых сообщений между объектами.

Последовательные диаграммы позволяют специфицировать временные требования к сообщениям и переходы объектов из одних состояний в другие под действием сообщений. Указанные возможности делают последовательные диаграммы особенно ценными при анализе требований к СРВ.

Описание сценария вызова лифта, выполненное в терминах последовательной диаграммы, представлено на рис. 3.3.

Сообщения могут быть помечены, как это сделано для первых двух сообщений рис. 3.3. Введение меток позволяет специфицировать временные ограничения, например, запись $\{b - a \leq 1\text{с.}\}$ означает, что между событиями b и a должно пройти не более 1с.

В случае необходимости последовательная диаграмма может быть дополнена состояниями объекта, как это сделано с состояниями «Ожидание» и «Действие» на рис. 3.3.

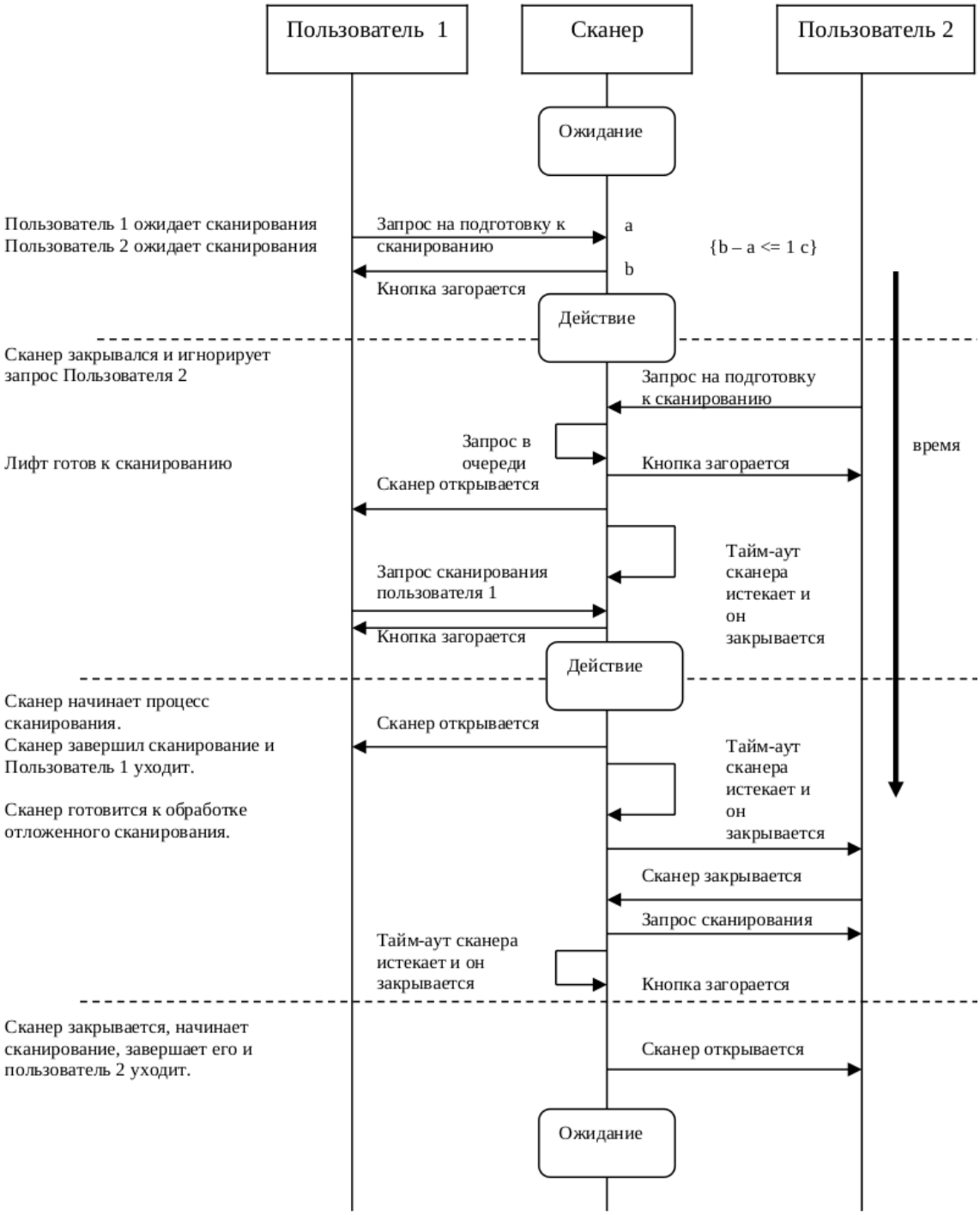

Рис. 3.3: Последовательная диаграмма, описывающая сценарий вызова сканера

Диаграмма сотрудничества показывает в основном ту же самую информацию, что и последовательная диаграмма, но акцентируется на статической структуре взаимодействующих объектов, а не на очерёдности сообщений или состояний.

Диаграмма сотрудничества системы лифта, описывающая тот же сценарий, что и рассмотренная выше последовательная диаграмма, представлена на рис. 3.4.

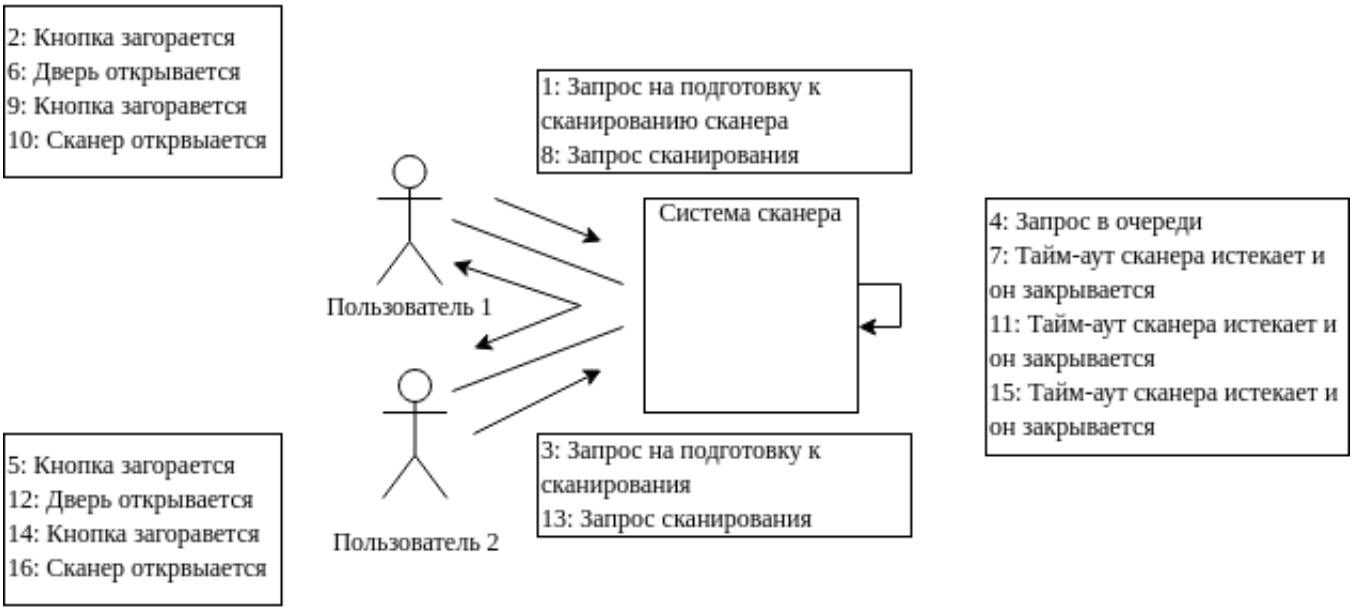

Рис. 3.4: Диаграмма сотрудничества системы сканера

4 Определение структуры системы

Составим список объектов, входящих в систему сканера. Для этого составим описание проблемы и подчеркнём существительные в этом описании.

Контроллер должен управлять сканером.

Сканер содержит набор кнопок подготовки к сканированию. Сканер имеет индикатор текущего сканирования на панели управления. Сканер имеет кнопку тревоги и Кнопка Пуск/ /Стоп.

Сканер имеет крышку, открывающуюся вертикально, которая закрывается при начале сканирования и закрывается при конце сканирования.

Сканер имеет датчики давления и оптический датчик для предотвращения закрытия сканера, когда препятствие находится между крышкой и сканером. Если препятствие обнаруживается одним из датчиков, крышка будет открыта.

Крышка будет автоматически закрыта после тайм-аута 5 секунд после открытия крышки. Обнаружение препятствия приводит к открытию крышки и повторно запускает тайм-аут.

У сканера есть панель управления с кнопкой для подготовки к сканированию, кнопкой тревоги и Кнопка Пуск/Стоп. Так же на панели отображается процесс сканирования.

Система должна откликаться на подготовку к сканирования, когда сканер свободен.

В других случаях запрос откладывается, пока сканер не отработает предыдущий запрос.

Кнопка подготовки к сканированию при нажатии подсвечивается, чтобы показать, что запрос принят. Нажатие кнопки подготовка к сканированию, когда такой запрос уже есть, не имеет эффекта.

Когда сканер заканчивает подготовку, чтобы обработать запрос, подсветка выключается.

Если кнопка подготовки к сканированию нажимается, когда сканер готов к сканированию, но он закрыт, то крышка открывается и запускается тайм-аут.

С целью обеспечения безопасности напряжение питания измеряется датчиком натяжения питания. В случае повреждения питания, когда измеренное напряжение падает ниже критического значения, датчик передает информацию в группу обслуживания, а аварийный блок питания поддерживает напряжение для того, чтобы сканер произвёл экстренное завершение работы без ошибок в сохранении.

В таблице 4.1 представлен обработанный перечень подчёркнутых существительных из приведённого описания проблемы.

Сканер	Тайм-аут
Кнопка начала сканирования	Подсветка кнопки
Кнопки запроса сканирования	Запрос сканирования
Крышка сканера	Контроллер сканера
Индикатор текущего сканирования	Панель управления сканером
Кнопка тревоги	Кнопка Пуск/Стоп
Датчик давления	Оптический датчик
Препятствие	Группа обслуживания
Напряжение питания	Датчик напряжения питания
Измеренное напряжение	Критическое значение
Аварийный блок питания	Экстренное завершение работы

Таблица 4.1: Список объектов системы сканера

После определения общего перечня объектов требуется определить перечень самых важных объектов системы.

Для случая с системой управления сканером к ним следует отнести следующие объекты:

1. Аварийный блок питания
2. Датчик напряжения питания
3. Сканер
4. Кнопка Пуск/Стоп
5. Кнопка начала сканирования
6. Кнопка тревоги
7. Индикатор текущего сканирования
8. Крышка сканера
9. Сканирование
10. Кнопка запроса сканирования
11. Датчик препятствия
12. Группа обслуживания
13. Контроллер сканера

Стратегии определения отношений между объектами в основном базируются на поиске объектов, обменивающихся между собой сообщениями.

В случае с системой управления сканером результат поиска объектов, обменивающихся сообщениями, представлен в таблице 4.2.

Источник сообщения	Получатель сообщения	Характер сообщения
Контроллер сканера	Сканер	Запрос статуса
Сканер	Группа обслуживания	Статус
Датчик напряжения питания	Группа обслуживания	Тревога
Датчик напряжения питания	Аварийный блок питания	Включить
Группа обслуживания	Аварийный блок питания	Отключить
Кнопка тревоги	Группа обслуживания	Тревога
Кнопка Пуск/стоп	Сканер	Пуск/стоп
Кнопка Пуск/стоп	Группа обслуживания	Пуск/стоп
Кнопка запроса сканирования	Контроллер сканера	Запрос сканирования
Контроллер сканера	Сканер	Добавить запрос
Кнопка начала сканирования	Контроллер	Добавить запрос
Датчик препятствий	Крышка	Открыть
Контроллер сканера	Индикатор	Статус сканирования

Таблица 4.2: Источники и получатели сообщений в системе сканера

Совокупность объектов и их отношений позволяет построить диаграмму объектов системы сканера. Диаграмма представлена на рис. 4.1.

Рис. 4.1: Диаграмма объектов системы сканера

Для определения наиболее важных атрибутов объекта в случаях, когда это не совсем очевидно, рекомендуется ответить на ряд вопросов:

1. Каковы функции объекта?
2. Какая информация необходима объекту для выполнения этих функций?
3. Над какой информацией выполняет объект свои операции?
4. Какой информацией о самом себе должен владеть объект?
5. Не являются ли идентифицированные атрибуты сами объектами?

Ответим на такие вопросы применительно к сканеру:

1. Сканировать объекты пользователей.
2. Для этого необходима информация о состоянии сканировании(остановлен, сканирует, занят, свободен), список запросов.
3. Сканер выполняет операции над той же самой информацией, которая представлена в ответе на вопрос 2.
4. Сканер должен знать своё состояние(остановлен, сканирует, занят, свободен) и свой процесс сканирования.
5. Сканер имеет крышку, которая имеет состояние(открыта, закрыта, открывается, закрывается) и имеет набор операций. Поэтому крышка является объектом. Аналогично для объекта ”Кнопка”.

Таким образом, сканер, как объект, обладает следующими атрибутами:

1. Состояние сканирования(сканирует, остановлен);
2. Состояние заполнение(занят, свободен);
3. Процесс сканирования;
4. Список запросов.

Среди объектов, определённых в описании проблемы, обычно существует много идентичных объектов.

В случае сканера, например, существует множество кнопок, но они все идентичны. Каждая кнопка активизируется при отпускании, прием запроса подтверждается подсветкой.

Объекты, которые идентичны по структуре и поведению, должны быть объединены в классы.

В системе сканера могут быть выделены следующие классы:

1. Отсек сканирования;
2. Сканер;
3. Контроллер;
4. Группа обслуживания;
5. Сканирование;
6. Индикаторы;
7. Крышка;
8. Кнопки;
9. Датчики;
10. Аварийный блок питания.

Для первой группы перечисленных классов будет создан единственный экземпляр:

1. Отсек сканирования;
2. Сканер;
3. Контроллер;
4. Крышка сканера;
5. Группа обслуживания.

Для второй группы будут отсутствовать классы-наследники:

1. Сканирование;
2. Индикаторы;
3. Аварийный блок питания.

Для третьей группы будут существовать классы-наследники. Диаграммы классов «Кнопки» и «Датчики» представлены на рис. 4.2 и рис. 4.3.

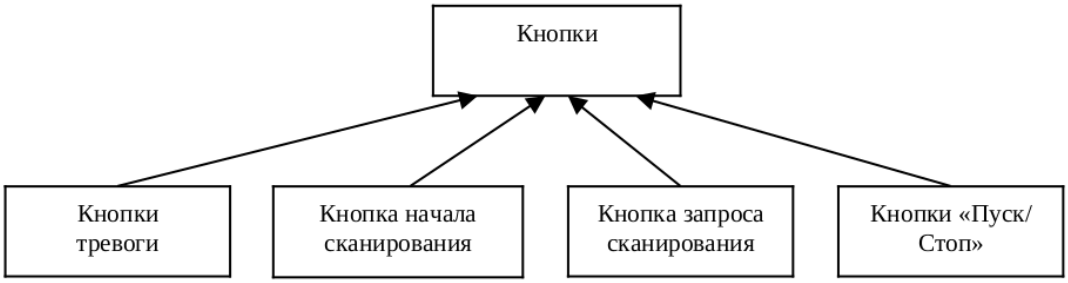

Рис. 4.2: Диаграмма классов «Кнопки»

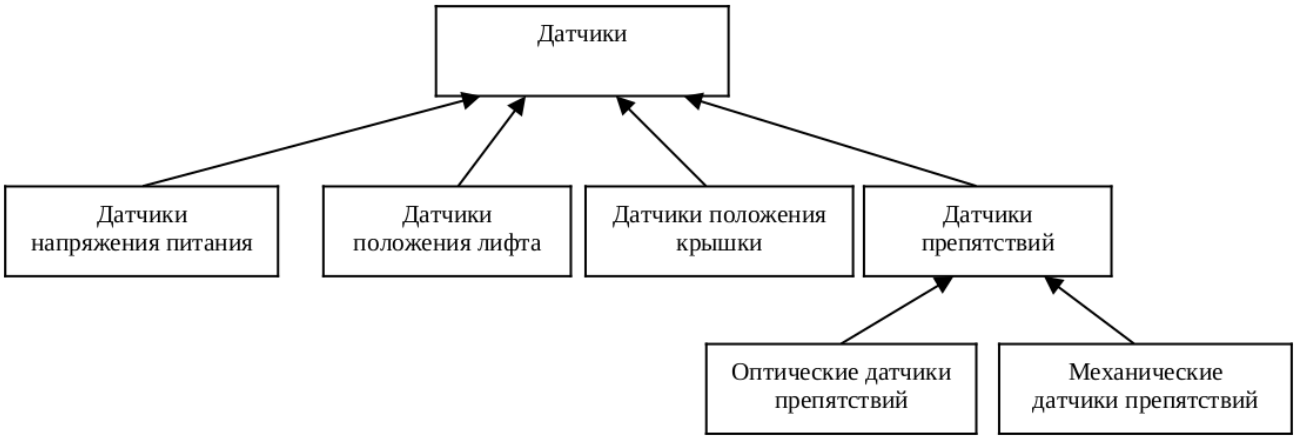

Рис. 4.3: Диаграмма классов «Датчики»

Первый вариант диаграммы классов системы сканера показан на рис. 4.4. Диаграмма показывает основные классы и их отношения. Вследствие ограниченных возможностей графического представления диаграммы классов, типы отношений и их множественность охарактеризованы словесно.

1. Отношение «Сканер — Аварийный блок питания» - блок питания находится в сканере (композиция, 1 — 1);

2. Отношение «Сканер — Датчик» — экземпляр класса «Сканер» — датчик натяжения питания находится в сканере (композиция, Много-к-одному);

3. Отношение «Аварийный блок питания — Контроллер» — блок питания управляется контроллером (ассоциация, 1 — 1);

4. Отношение «Контроллер — Группа обслуживания» — контроллер передает информацию группе обслуживания, группа обслуживания управляет системой лифта через контроллер (ассоциация, 1 — 1);

5. Отношение «Датчик — Контроллер» — экземпляры класса «Датчик» передают информацию контроллеру (ассоциация, Много-к-одному);

6. Отношение «Контроллер — Сканер» — контроллер управляет сканером (ассоциация, 1 — 1);

7. Отношение «Крышка — Сканер» — экземпляры класса «Крышка» находятся в сканере (композиция, 1 — 1);

8. Отношение «Крышка — Контроллер» — экземпляр класса «Крышка» управляется контроллером (ассоциация, 1 — 1);

9. Отношение «Кнопка — Сканер» — экземпляры класса «Кнопка» находятся в сканере (композиция, Много-к-одному);

10. Отношение «Индикатор — Сканер» — индикатор процесса сканирования находится в сканере (композиция, 1 — 1);

11. Отношение «Кнопка — Контроллер» — экземпляры класса «Кнопка» передают нажатия контроллеру, а контроллер передает команду «подсветить» кнопкам (ассоциация, Много-к-одному);

12. Отношение «Индикатор — Контроллер» — индикаторы получают информацию от контроллера (ассоциация, Много-к-одному).

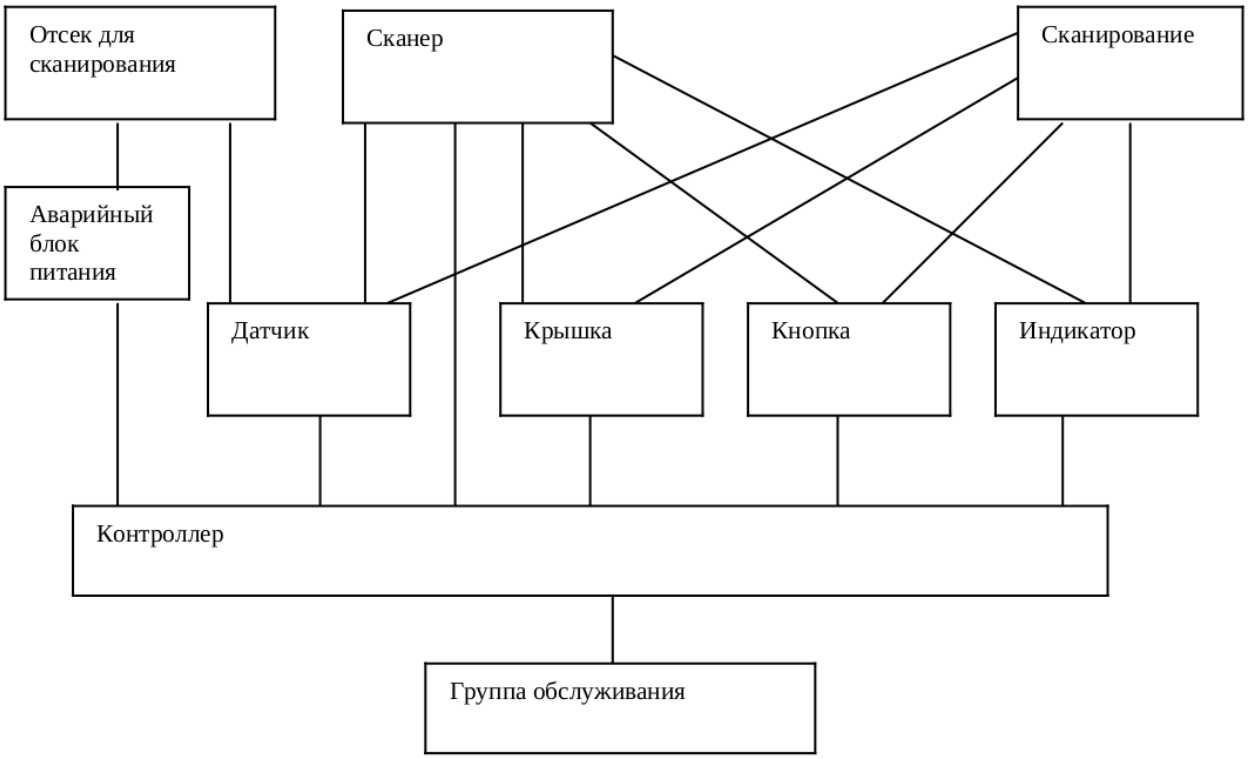

Рис. 4.4: Первый вариант диаграммы классов сканера

5 Описание поведения системы

В предыдущем разделе были определены базовые объекты системы, классы и их отношения. Опишем теперь поведение системы.

Самым важным средством описания поведения системы является описание с помощью диаграмм состояний.

Другим средством является описание сценариев, которые уже были использованы на этапе анализа требований к системе

Сканер представляет собой систему с большим числом возможных состояний. При этом возможна вложенность состояний (подсостояния), параллельность и многоаспектность понятия состояния.

На первом этапе выделим два возможных состояния сканера:

1. Свободен;
2. Занят.

Будем считать, что сканер находится в состоянии «Свободен», если выполняются следующие условия:

1. Сканер не сканирует;
2. Отсутствие пользователей сканера;
3. Крышка сканера закрыта.

Указанные три условия позволяют выделить три подсостояния занятого сканера:

1. Сканирование — покой;
2. Отсутствие пользователей — Наличие пользователей;
3. Открытая крышка — Закрытая крышка.

Подсостояния 1 и 2, а также подсостояния 2 и 3 полностью ортогональны, т.е. независимы друг от друга. Сканирование или Покой могут быть как при Отсутствии, так и при Наличии пользователей. Открытая или Закрытая крышка может быть как при Отсутствии, так и при Наличии пользователей. Сканирование может иметь место только при Закрытой крышки.

Приведенные особенности поведения сканера следующим образом отображаются на диаграмме состояний сканера, рис. 5.1.

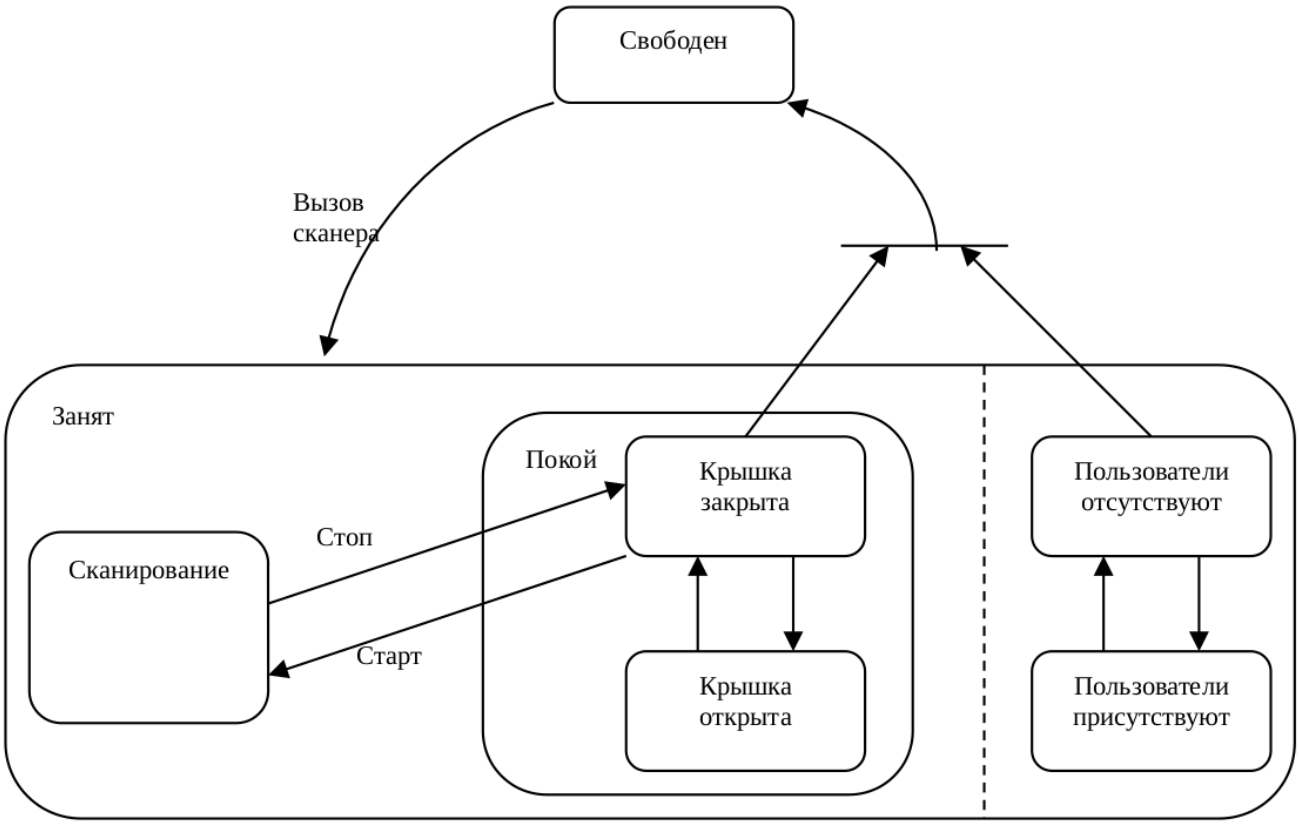

Рис. 5.1: Диаграмма состояний сканера

Диаграммы активности контроллера сканера при выполнении запроса сканера, когда сканер находится в состоянии «Свободен», представлены на рис. 5.2. Ясно, что такой элемент диаграммы, как, например, «Ожидание сканирования», сам может представлять собой целую последовательность действий.

Кроме того, диаграмм, аналогичных рис. 5.2, может целый ряд, в зависимости от того, в каком состоянии находился сканер, когда делался запрос.

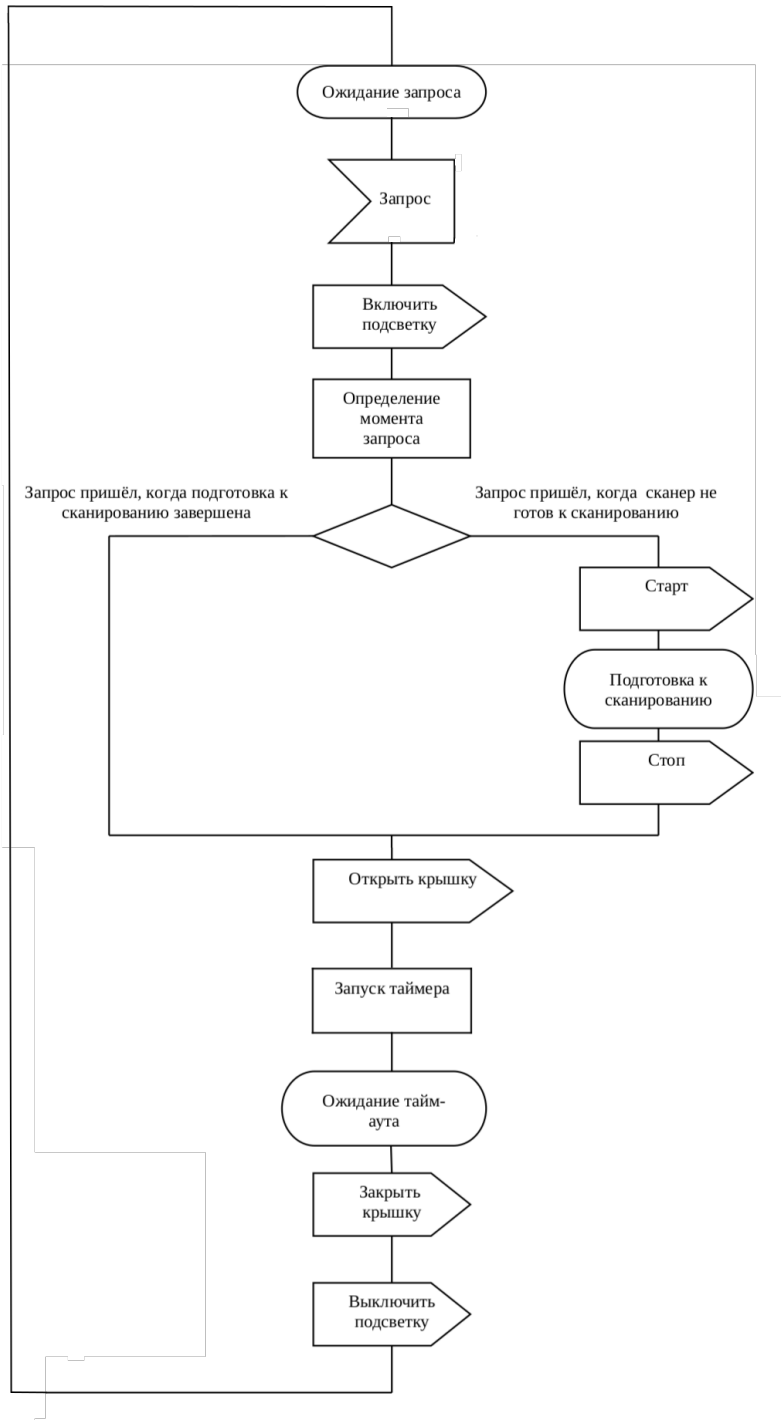

Рис. 5.2: Диаграмма активности сканера

Определим примерный набор операций классов Сканер, Крышка и Контроллер проектируемой системы.

Количество операций, также как и атрибутов, определяется степенью проработанности класса.

Например. Если различать подсостояния (Пользователи отсутствуют; Пользователи присутствуют), то необходимо ввести атрибут, имеющий смысл «Загружен» (”Loaded”). Если количество людей, пользующихся сканером, не различать, то атрибут Loaded может иметь тип Boolean, а если различать (например, чтобы формировать сигнал тревоги при перегрузке), то атрибут Loaded должен иметь тип Integer.

То же самое относится к включению в класс Сканер объектов Крышка, Кнопки.

```
TScannerState = (scannning, stop);
TProcess = (1..100);

TScanner = object
        State : TScannerState;
        Position : TProcess;
        Constructor Create;
        Destructor Done;
        Procedure Start;
        Procedure Stop;
        Procedure Scan;
        Function GetScanProcess : TProcess;
        Function GetState : TScannerState;
End;
```

Операции Start и Stop носят характер команд, которые контроллер посылает сканеру.

Операции GetScanPosition, GetState носят характер команд, которые контроллер посылает сканеру для чтения его состояния и процесса сканирования. Операция Scan – базовая операция сканера в данной модели.

```
TDoorState = (opened, closed, opening, closing);

TDoor = object
        State : TDoorState;
        Constructor Create;
        Destructor Done;
        Procedure Open;
        Procedure Close;
        Procedure Move;
        Function GetState : TDoorState;
End;
```

Операции Open и Close носят характер команд, которые контроллер посылает крышке. Операция GetState носит характер команды, которую контроллер посылает крышке для чтения ее состояния. Операция Move реализует «процесс» открытия или закрытия двери.

```
TEventType = (Alarm, ButtonPressed);
TButtonType = (LiftButton, FloorButton);


TEvent = record
        EventType : TEventType;
        ButtonType : TButtonType;
        Message : Integer;
End;

TController = object
        Constructor Create;
        Destructor Done;
        Function GetEvent : TEvent;
        Procedure Dispatcher;
End;
```

Операция Dispatcher носит характер «бесконечного цикла», состоящего из операции GetEvent в начале итерации цикла и, в зависимости от события, посылки соответствующих команд соответствующим объектам.

6 Архитектурное проектирование системы

К рассматриваемой системе сканера целесообразным выбором архитектурного образца является образец «Ведущий-Ведомый».

Ведущим (клиентом) будет выступать Контроллер, заявки от которого поступают к Сканеру, который будет выступать в качестве Ведомого (сервера).

Указанному образцу будем следовать при формировании облика физической архитектуры и распределении задач по процессорам этой архитектуры.

Типичная система реального времени имеет множество потоков управления, выполняющихся одновременно.

В системе сканер такие объекты, как Сканер, Контроллер, крышка, Кнопка, Датчики, могут порождать события, и в этом смысле являются активными объектами системы.

Активные объекты системы в своем составе имеют методы, выполнение которых происходит одновременно. Выявление этих методов и позволяет определить параллельные задачи в системе.

Представление параллельных задач в системе может быть выполнено путем трансформации диаграммы классов в диаграмму задач.

Диаграмма задач системы лифта представлена на рис. 6.1.

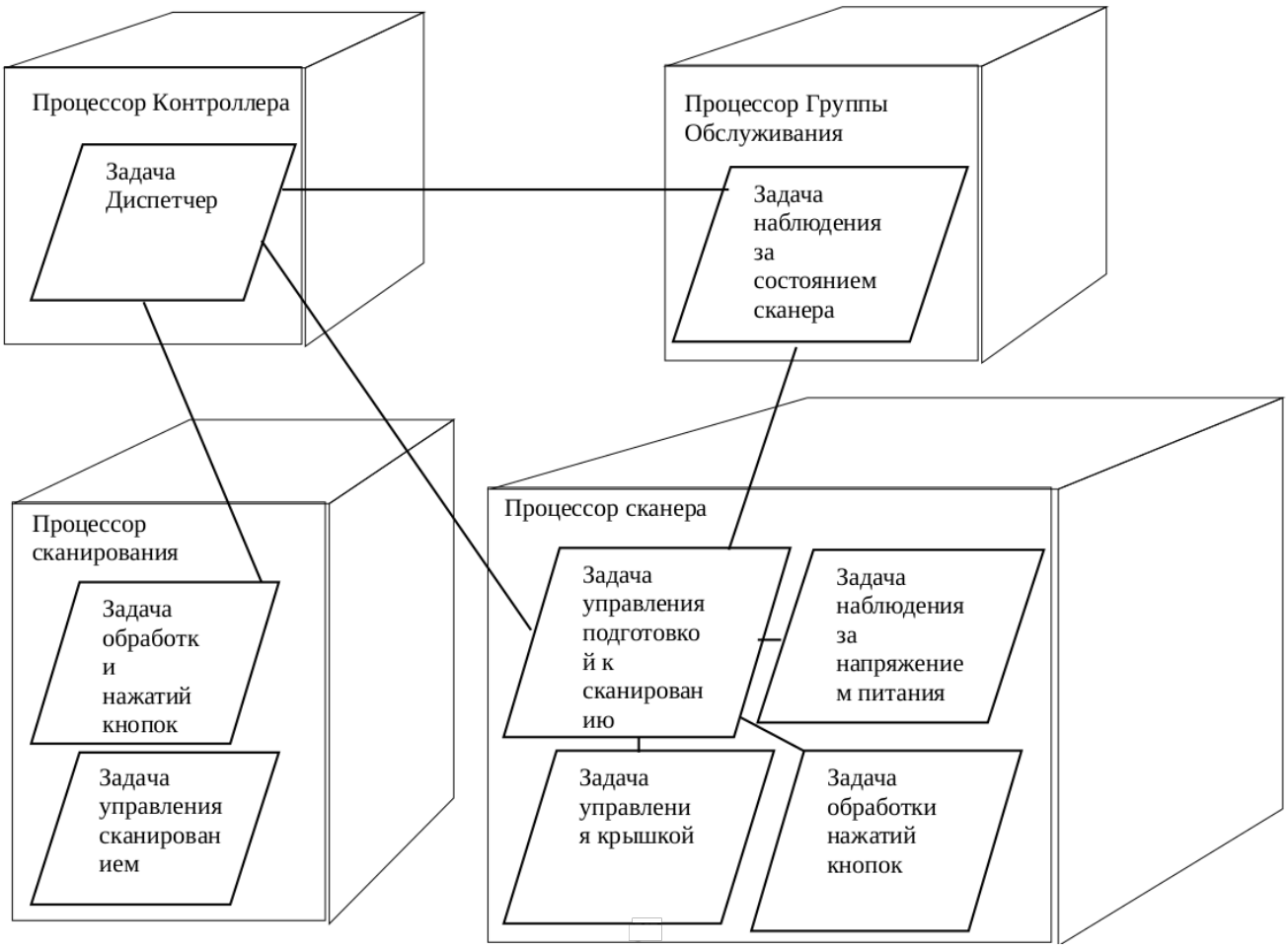

Рис. 6.1: Диаграмма задач системы сканера

Перед построением диаграммы задач были определены процессоры. Отметим, что процессоры контроллера, сканера и группы обслуживания – это одиночные процессоры.

7 Техническое проектирование системы

На предыдущем этапе проектирования – на этапе архитектурного проектирования были приняты решения относительно базовых аппаратных и программных компонентов.

Базовыми аппаратными компонентами выступали процессоры, а программными классы, объекты и задачи.

Итогом предыдущего этапа проектирования явилась диаграмма задач системы.

В данном подразделе дадим характеристику среднего этапа проектирования, который назовём техническим.

На данном этапе к базовым классам добавляются вспомогательные классы для поддержки функционирования базовых классов.

Существует набор образцов для данного этапа проектирования, некоторые из которых, как наиболее важные для систем реального времени, представлены в таблице. 6.1.

Категория	Имя образца	Цель
Простые образцы	Наблюдатель и Транзакции	Наблюдатель позволяют группе клиентов разделять сервер. Транзакции позволяют управлять коммуникациями между объектами.
Повторно используемые образцы	Контейнер, Интерфейс и Рандеву	Контейнер позволяет агрегировать объекты и эффективно реализовывать ассоциации типа «один ко многим». Интерфейс обеспечивает отделение типа объекта от его реализации с целью поддержки множественной реализации данного типа. Рандеву обеспечивает гибкий механизм для межзадачных коммуникаций.
Образцы управления состояниями	Состояние и Таблица состояний	Состояние обеспечивает реализацию машины состояний при редких изменениях состояний. Таблица состояний обеспечивает эффективные средства поддержки машин с большим числом состояний.

Таблица 7.1: Образцы технического этапа проектирования

Применительно к рассматриваемой системе сканере будет использоваться образец «Контейнер» при реализации ассоциаций между Контроллером и Сканированиями. Так, информацию о текущем сканировании объекта Сканер в Отсеке для сканирования необходимо рассылать на все сканирования и целесообразно это делать одной операцией – итератором, входящей в класс Контейнер.

8 Детальное проектирование системы

В общем случае на данном этапе определяются решения относительно следующих характеристик объектов:

1. Структуры данных;
2. Реализация ассоциаций;
3. Множество операций, определённых на данных;
4. Видимость данных и операций;
5. Алгоритмы, используемые для реализации этих операций;
6. Возбуждаемые и обрабатываемые исключения.

Структуры данных в объектах обычно просты, поскольку, если это не так, то целесообразно создать отдельный объект, чтобы хранить такие данные. Однако, необходимо определить не только структуры данных, но и диапазоны, точность, предусловия, начальные значения. Это необходимо сделать, поскольку проектируемые системы имеют дело с реальными объектами, которые должны функционировать без ошибок. Так, например, применительно к системе сканер, процесс текущего сканирования может быть описан типом integer, а можно ограничить этот атрибут диапазоном значений $[1..N]$, где N — максимальное количество сканирований.

Реализация ассоциаций позволяет объектам-клиентам активизировать операции объектов-серверов. Простейший способ реализации ассоциации – это прямой вызов операции сервера. Однако, такой способ будет неприемлем, если объекты выполняются в различных потоках, и тем более, на различных процессорах. В этих случаях необходимо использовать средства ОС, такие как очереди сообщений и удаленные вызовы процедур. Применительно к системе сканера (с учетом реализации программных моделей объекта Сканер и объекта Контроллер) метод Сканера, реализующий сканирование, и метод Контроллера, реализующий прием и диспетчеризацию событий, должны выполняться в разных потоках. Поэтому необходима реализация средств взаимодействия потоков, таких как, например, буфер или очередь сообщений.

Операции, определённые на данных. Архитектурное проектирование специфицирует только базовые операции объектов, доступные из внешнего окружения. Детальное проектирование добавляет еще и внутренние операции, которые появляются в результате функциональной декомпозиции базовых операций.

Видимость данных и операций. Существует несколько принципов управления видимостью данных и операций, а именно:

1. Видимыми надо делать только те данные и операции, которые необходимы объектам-клиентам;
2. Видимые операции должны быть семантически понятными;
3. Атрибуты объекта не должны быть видимы объектам-клиентам.

Алгоритмы, используемые для реализации операций. Существует ряд критериев, по которым можно оценивать качество разрабатываемых алгоритмов. Примерами таких критериев являются следующие критерии:

1. Производительность (средняя и рассчитанная на наихудший случай);
2. Требования памяти;
3. Простоты;
4. Усилия разработки;
5. Повторная используемость;
6. Расширяемость;
7. Надёжность.

Перечисленные критерии конфликтны по своей сути. Выбор критериев определяется характером системы. Так, для жестких систем реального времени важным может стать критерий производительности, рассчитанной на наихудший случай. А если СРВ является еще и встроенной системой, то важными могут оказаться и требования к памяти. Для описания алгоритмов может быть использован структурированный английский язык, псевдокод, а также диаграммы активности, рассмотренные выше.

Возбуждаемые и обрабатываемые исключения. Обработка исключений является существенным усложняющим фактором при проектировании алгоритмов. В минимальном варианте реализации алгоритма необходимо специфицировать исключения, которые необходимо обрабатывать, и исключения, которые необходимо возбуждать. Общий подход заключается в том, чтобы обрабатывать те исключения, для обработки которых имеется достаточно информации, а возбуждать все остальные исключения. Серьезность ситуации, которая приводит к исключению, может определить место, в котором это исключение обрабатывается. Так, в случае со сканером, резкое изменение напряжения питания должно приводить систему в безопасное состояние на уровне системы в целом, а не на уровне отдельных объектов.

9 Описание реализации программной модели системы

Реализация системы должна включать набор объектов (программных моделей реальных объектов), связанных отношением клиент-сервер. Например, для системы управления сканер такими объектами являются Сканер и Контроллер. Объекты, выбираемые для реализации, должны иметь в своем составе методы, выполняемые как потоки.

С добавлением других объектов растет степень проработанности системы, а, следовательно, и качество курсового проекта. Например, включение в объект Сканер объекта Крышка позволяет более детально смоделировать работу системы.

Активизация событий в программной модели системы должна выполняться специально выделенными для каждого события клавишами. Например, клавиши ”1” — ”9” могут имитировать кнопки запроса сканирования с соответствующих этажей.

10 Описание реализации ядра управления задачами

Для реализации многопоточной программы в однозадачной операционной системе следует создать ядро управления задачами.

Примерный состав примитивов ядра следующий:

1. Создать ядро;
2. Уничтожить ядро;
3. Начать работу ядра;
4. Завершить работу ядра;
5. Создать задачу;
6. Уничтожить задачу;
7. Приостановить задачу;
8. Возобновить задачу;
9. Создать семафор;
10. Уничтожить семафор;
11. Выполнить операцию $P()$ семафора;
12. Выполнить операцию $V()$ семафора;

Ниже приведены описания объектов Сканер, Семафор и Ядро.

```
#include <setjmp.h>
#include <dos.h>

class TScanner : {
        int stack[1000];
        jmp_buf ts;
public:
TScanner ( void *p ) {
                struct SREGS segs;
                segread ( &segs );
                ts[0].j_sp      =       FP_OFF(stack) + sizeof stack;
                ts[0].j_ss      =       FP_SEG(stack);
                ts[0].j_flag    =       0x200;
                ts[0].j_cs      =       FP_SEG(p);
                ts[0].j_ip      =       FP_OFF(p);
                ts[0].j_bp      =       ts[0].j_sp;
                ts[0].j_di      =       0;
                ts[0].j_es      =       segs.es;
                ts[0].j_si      =       0;
                ts[0].j_ds      =       segs.ds;
        }
};



class TSemaphore {
        int count;
        TCollection *List;
public:
        TSemaphore();
        ~TSemaphore();
        void P();
        void V();
};
```

```
#include <tv.h>
#include "proc.h"
#include "sema.h"

class TCore{
        TCollection     *ReadyList;
        TCollection     *KillList;
        TCollection     *DelayList;
        TProcess        *cur;
        unsigned long count;
        void swt(jmp_buf from,jmp_buf to);
public :
        TCore();
        ~TCore();
        void start();
        void stop();
        void delay(unsigned long tik);
        void resume();
};
```

Представленные тексты программы показывают реализацию ядра.

11 Список литературы

1. ”Теория и методы автоматизации проектирования вычислительных систем” — Брейер Мелвин А.

2. ”Автоматизация проектирования вычислительных систем. Языки, моделирование и базы данных” — ред. Брейер М.

3. ”Разработка встроенных систем с помощью микроконтроллеров PIC. Принципы и практические примеры” — Тим Уилмсхерст

4. ”Проектирование встраиваемых систем на ПЛИС” — Заиналабедин Наваби

5. ”Встраиваемые системы. Проектирование приложений на микроконтроллерах семейства 68HC12 / HCS12 с применением языка C” — С.Ф. Баррет

6. ”Встраиваемые системы” — Барретт С.Ф., Пак Д.Дж.

7. ”Паттерны проектирования” — Эрик Фримен, Элизабет Фримен

8. ”Объектно-ориентированны анализ и проектирование с примерами приложений. Третье издание” — Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон

9. ”Предметно-ориентированные языки программирования. М.:"Вильямс 2011.” — Мартин Фаулер (при участии Ребекки Парсонс).

10. ”An Embedded Software Primer” — Simon D.

11. ”Programming With GNU Software” — Mike Loukides, Andy Oram

12. ”Making Embedded Systems: Design Patterns for Great Software” — Elecia White