Глава 4
Системная архитектура

Предисловие

В этой главе мы будем ориентироваться в основном на проекты, в которых выполняются системный анализ и проектирование – т.е. на проекты по разработке систем, вкоторых задействованы несколько инженерных дисциплин. Как правило, такие системы состоят из программных, электронных, механических и химических компонентов. Системная архитектура определяет требования к системе в целом, а также архитектуру системы, состоящей из подсистем, которые в свою очередь состоят из компонентов, создаваемых различными инженерными дисциплинами. В предыдущей главе мы обсуждали сбор и анализ требований к системе. В этой главе мы рассмотрим спецификацию архитектуры уровня системы.

Многие проекты ориентируются в основном на разработку только программного обеспечения; в этом случае не требуется затрачивать серьезных усилий на спецификацию системной архитектуры, а большая часть усилий должна быть затрачена на спецификацию программной архитектуры (об этом будет рассказываться в главе 6). Однако даже для проектов, ориентированных на разработку программного обеспечения, эта глава будет полезной, поскольку в ней рассказывается о том, что же представляет собой системная архитектура, и рассматривается один из ее важнейших аспектов – определение подсистем. Мы рассмотрим один из важнейших составляющих этого процесса – декомпозицию вариантов использования уровня системы на варианты использования уровня подсистем, а также спецификацию интерфейсов и протоколов взаимодействий между подсистемами.

Я хотел бы еще раз напомнить, что мы подразумеваем под термином *архитектура*. Архитектура включает в себя высокоуровневые проектные решения, которые влияют на большую часть или на всю систему в целом. Заметим, что разработка архитектуры является составной частью проектирования. В процессе Harmony анализ состоит в спецификации основных аспектов системы – ее свойств и характеристик, которые необходимы для выполнения системой своего назначения. Проектирование, в свою очередь, сфокусировано на оптимизации модели анализа в соответствии с так называемыми *критериями проектирования* - набором аспектов системы, в соответствии с которыми различные проектные решения и выбор технологий могут быть измерены, оценены и в результате выбраны. Критерии проектирования могут относиться к производительности системы (такими как производительность в наихудшем случае, ширина полосы пропускания, пропускная способность), использованию системных ресурсов во время работы (таких как память), метрикам качества времени проектирования (например, коэффициент сложности и инкапсуляции), характеристикам результатов проектирования (например, удобство сопровождения, возможность повторного использования или портирования), или даже к проектным характеристикам (таким как, требуемые ресурсы).

Процесс Harmony определяет три уровня проектирования. Архитектурное проектирование нацелено на оптимизацию системы в целом при помощи применения согласованных друг с другом архитектурных решений. Техническое проектирование нацелено на оптимизацию кооперации объектов, взаимодействующих друг с другом с целью реализации определенной возможности системы (например, отдельного варианта использования). При детальном проектировании выполняется оптимизация отдельных объектов. В этой главе мы сконцентрируем свое внимание только на архитектуре, поскольку именно она является целью системного проектирования.

Процесс Harmony определяет пять ключевых представлений архитектуры (см. Рис. 4-1)

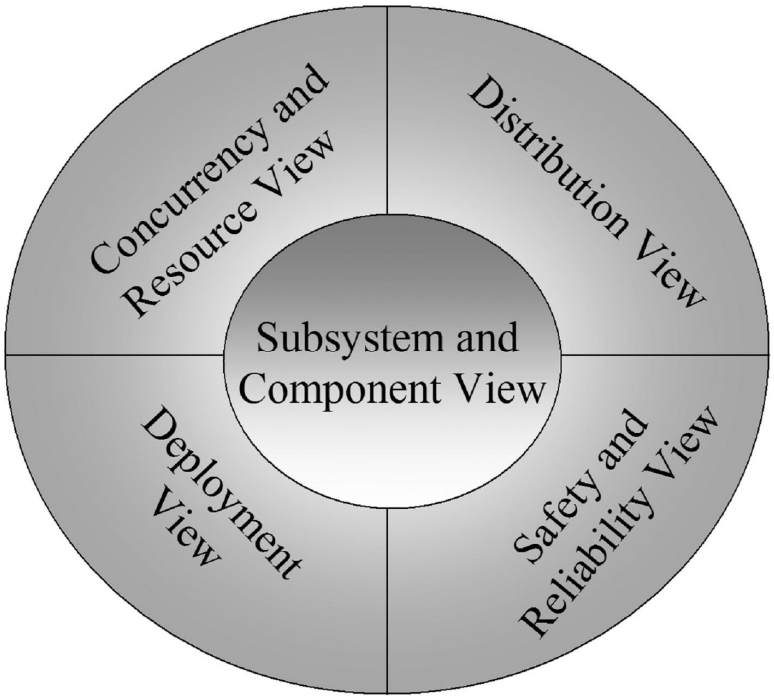

Рис. 4-1 Представления архитектуры

Представление параллелизма и ресурсов определяет аспекты системы, связанные с параллельным исполнением, а также каким образом паралельные потоки будут использоваться совместно используемые ресурсы. Представление развертывания определяет каким образом объекты будут развернуты по различным адресным пространствам (например, по процессорам), и как будет реализовано взаимодействие между ними эффективным и надежным образом . Представление безопасности и надежности определяет каким образом выполняются задачи идентификации, изоляции и реакции на различные отказы по время работы системы. Представление размещения определяет какие инженерные дисциплины будут реализовывать элементы системы и как они будут взаимодействовать. И наконец, представление подсистем и компонентов имеет дело с определением составных частей системы самого высокого уровня, а также как распределяется функциональность системы между этими частями и как эти части взаимодействуют друг с другом на прикладном уровне. На этом последнем представлении архитектуры и сфокусировано главным образом системное проектирование, хотя, разумеется, в нем могут затрагиваться и другие аспекты системы.

Под *системной архитектурой* мы понимаем определение стратегических проектных решений, которые влияют на большую часть или на всю систему *с системной точки зрения*. Системная точка зрения находится "над" программным обеспечением, электроникой и механикой. Таким образом, системная архитектура будет сосредоточена главным образом на спецификации набора подсистем, на которые разбивается проектируемая система, и привязке требований и функциональности к данным подсистемам, а также определении интерфейсов между этими подсистемами. По окончанию деятельности по разработке системной архитектуры отдельные части системной модели передаются командам, занимающихся разработкой подсистем. Последние разобьют подсистемы на компоненты из различных инженерных дисциплин, после чего начнется их более детальный анализ и проектирование.

До того как это произойдет, разработка системной архитектуры в основном сосредоточена на представлении подсистем и компонентов, хотя время от времени она также может "углубляться" в другие представления.

Задание 4.1 Организация модели системы

Об организации модели системы члены команды редко задумываются до тех пор, пока не возкникнут проблемы по управлению сложностью разрабатываемой системы. Реорганизация модели в этот момент потребует нетривиальных и незапланированных усилий. Проекты, в которых есть выделенные системные инженеры, выполняющие системный анализ, как правило достаточно сложны, поэтому уже на ранних стадиях необходимо определить как будет организовано управление моделью.

Существует много различных способов организации моделей. В этой книге мы рассмотрим только один из них, хотя существуют и другие. Как всегда, существует множество *плохих* способов организации моделей, равно как и множество хороших способов. Вот список вопросов, которые необходимо принять во внимание при организации структуры моделей:

- Большой или маленький проект?
- Одна модель или несколько моделей?
- Один продукт или несколько продуктов?
- Как общие части будут использоваться совместно всеми участниками проекта?
- Каким образом архитектурные решения будут доступны участникам проекта?
- Как гарантировать удовлетворение архитектурным решениям всеми участниками проекта?
- Будут ли все участники проекта находиться в одном месте, или проектная команда будет распределенной?
- Как минимизировать расходы на управление моделью?
- Какие принципы будут использоваться для группировки элементов друг с другом?
- Каким образом будет организовано конфигурационное управление?

Один из первых вопросов: стоит ли создавать одну большую модель, единую для всех членов команды, или же несколько различных моделей. Преимуществом одной модели является простота управления – существует только одна сущность которой необходимо управлять (даже если этот объект имеет вложенные части). С другой стороны, использование одной модели приводит к увеличению времени ее загрузки, что необходимо учитывать, если число участников проекта составляет 50 или более человек, а также усложнение поиска различных элементов для совместного использования и взаимодействия. Преимущество нескольких моделей состоит в том, что сложность системы может быть распределена по нескольким моделям, каждая из которых меньше по объему, легче управляется, и требует меньше времени для загрузки, чем одна большая модель. С другой стороны, необходимо очень хорошо подумать как произвести разбиение на модели, какие критерии должны использоваться для размещения элементов по различным моделям, а также как модели будут совместно использоваться участниками проекта.

В качестве рекомендации: для проектов с числом участников не более 15 человек подходит вариант одной модели; для проектов, в которых участвуют более 20 человек, создание несколько взаимодействующих моделей может быть лучшим выбором. Разумеется, на решение использовать одну или несколько моделей влияют свойства разрабатываемой системы. Если модель линейно разделима (т.е. ее можно разделить на некоторое число более или менее независимых друг от друга частей), управление несколькими моделями не составляет труда. Если система не является линейно разделимой, то создание нескольких моделей может оказаться гораздо более сложным делом. Если для работы с одной моделью необходимо загружать и все остальные модели, разбиение системы на несколько моделей не будет вам полезно.

Главные причины разбиения больших моделей следующие: 1) уменьшить время, необходимое для загрузки и сохранение и 2) предоставить меньшие, но достаточные модели для отдельных команд, имеющих более узкий фокус. Именно на эту вторую причину мы будем ориентироваться далее . Основные группы лиц, имеющие дело с системной моделью, перечислены в Таб. 4-1 .

Таб.4-1 Группы лиц, использующих системную модель

Группа	Назначение	Область интересов
Системные инженеры	• Создание модели требований и архитектуры, независимых от инженерных дисциплин • Привязка требований и функциональности к подсистемам • Определение интерфейсов между подсистемами	• Системные требования • Системная архитектура
Команда разработки подсистемы	• Определение архитектуры размещения для отдельной подсистемы • Создание спецификации на программное и аппаратное обеспечение (вложенные модели)	• Отдельная подсистема • Распределение требований к подсистеме по отдельным инженерным дисциплинам
Инженеры (разработчики программного обеспечения, конструктора, электронщики, химики)	• Проведение анализа и проектирования определенной подсистемы в рамках своей специализации	• Элементы модели подсистемы, относящиеся к их инженерной дисциплине
Архитектор проекта	• Определение проектной архитектуры для множества подсистем	• Общая архитектурная модель для всей системы • Архитектуры подсистем для каждой инженерной дисциплины
Тестировщики	• Проведение тестирования подсистем • Проведение интеграционного тестирования • Проведение приемочного тестирования	• Подсистема – как правило, производится внутри команды, разрабатывающей подсистему • Интеграция тестирование – как правило, выполняется на уровне всей системы в целом • Приемка – как правило, выполняется на уровне всей системы в целом

Для небольших систем мы рекомендуем следующую организацию модели:

- Системная область
  - Системные требования
    - Операционные требования (варианты использования)
    - Неоперационные требования
    - Тестовые сценарии для системы
  - Системная архитектура (включая размещение)
- Область сборки системы (для поэтапной сборки системы)
  - Сборка x (отдельный пакет для каждой сборки)
- Общая область (для совместно используемых типов и классов)
- Кооперация x (отдельный пакет для каждого варианта использования)

Основным преимуществом предложенной организации модели является ее простота, однако к ее недостаткам следует отнести плохую масштабируемость для больших команд и больших проектов. Для средних и крупномасштабных систем мы рекомендуем следующую организацию модели, позволяющая учесть интересы и потребности всех участников проекта.

- Системная область
  - Системные требования
    - Операционные требования (варианты использования)
      - Вариант использования x (для каждого варианта использования)
        - "Черный ящик"
        - "Белый ящик"
    - Неоперационные требования
    - Тестовые сценарии для системы
  - Системная архитектура
- Область сборки системы (для поэтапной сборки системы)
  - Сборка x (отдельный пакет для каждой сборки)
- Общая область (для совместно используемых элементов)
  - Интерфейсы подсистем
  - Совместно используемые предметные области
- Область подсистемы x (для каждой подсистемы)
  - Требования к подсистеме
    - Операционные требования к подсистеме (варианты использования)
      - Вариант использования x (для каждого варианта использования подсистемы)
    - Неоперационные требования
    - Тестовые сценарии для подсистемы
  - Архитектура подсистемы
    - Размещение для подсистемы
    - Междисциплинарные интерфейсы (например, интерфейсы между ПО и электроникой, электронными и механическими компонентами)
  - Кооперация x (отдельный пакет для каждого варианта использования)

Данная организация может использоваться для моделей от средних до больших размеров. Для проектов среднего размера можно не создавать пакеты "Черный ящик" и "Белый ящик" для варианта использования и хранить все элементы для варианта использования в одном пакете. Для небольших моделей может оказаться достаточным одного пакета “Требования”, который содержит все необходимую виды требований (операционные (варианты использования), неоперационные, элементы текстовых требований, ограничения и т.д.). Данная организационная схема может использоваться внутри одной модели, но она также содержит точки для ее разделения на несколько моделей. Для проектов больших систем мы рекомендуем выносить следующие области в отдельные модели:

- Системная модель
- Общая модель
- Модели подсистем (одна для подсистемы)

В UML элементом организации модели является *пакет*. Пакет является элементом модели, который содержит другие элементы модели, определяя для них общую область видимости, в том числе содержат вложенные пакеты. Организация модели может быть показана либо на диаграмме пакетов (диаграмме классов, целью которой является показать организацию пакетов в одной или нескольких моделях [1])) или в представлении обозревателя модели.

Для этого первого задания:

1. Создайте единую модель для системы управления дорожным перекрестком Роадраннер, организовав ее так, как описано выше. Поскольку это проект среднего размера, разместите анализ всех вариантов использования в одном пакете.
2. Создайте набор моделей для системы БЛА Койот, используя предложенную выше организацию для крупномасштабного проекта. Поскольку это модель большой системы, создайте отдельные пакеты для анализа вариантов использования уровня системы, со вложенными пакетами анализа "черного ящика" и "белого ящика".

Так как подсистемы и домены еще не определены, вы можете оставить пустой области подсистем в вашей модели. Мы уточним данную область организации модели после того как определим подсистемы и предметные области чуть далее в этой главе..

Задание 4.2 Определение подсистем

На данный момент мы определили общую организацию модели ("логическую архитектуру") системы. Это может быть одна модель – как для системы управления дорожным перекрестком Роадраннер – или нескольких связанных друг с другом моделей – как в случае системы БЛАК. Это означает, что мы определили набор организационных единиц, которые существуют во время разработки и позволяют управлять сложностью проекта. Точно также нам необходимо определить организацию элементов системы во время ее работы. Пакеты, используемые для

[1] Модель выполняет роль аналогичную пакету.

организации моделей, существуют лишь во время разработки. Пакеты – это элементы для которых не создаются работающие экземпляры, поэтому они служат только для организации элементов модели.

Для организации элементов, существующих во время работы системы (т.е. объектов), нам необходимо использовать элементы для которых могут быть созданы экземпляры - аналогичные объектам. В UML это означает создание экземпляров классов. UML определяет две разновидности классов, используемых для высокоуровневой организации системы во время работы. А именно, для этих целей используются подсистемы и компоненты, однако по своей сути, они являются просто структурированными классами. Структурированный класс – это просто класс, который содержит внутренние части, специфицируемые также классами. Подсистемы и компоненты по сути являются структурированными классами большого масштаба. Между ними существует несколько технических различий, таких как связь компонента с артефактами, однако эти различия очень незначительны. Под артефактом мы имеем здесь ввиду физическую реализацию сущности во время работы – например, файл .DLL, .EXE, или .LIB, определяющие соответствующий объект. Подсистема является частным случаем компонента. В действительности, все они аналогичны классам.

Хотя UML определяет такие элементы, как класс, система, компонент и подсистема, в UML явно не указано, каким образом они должны использоваться. В процессе Harmony рекомендуется придерживаться определенному набору принципов, которые оказались для этого эффективными:

- Система – это класс, который содержит в себе наиболее крупные элементы в вашем проекте.
- Подсистема – это класс, который является составной частью системы первого уровня вложенности (то есть подсистемы – это составные части системы).
- Компонент - это класс уровня программного обеспечения, который является составной частью элемента подсистемы первого уровня вложенности.
- Задача – это структурированный класс, который содержится внутри компонента и является активным (т.е. управляет отдельным потоком). Задача – это основной элемент параллелизма в модели.

На практике, работающие системы организуются в крупномасштабные объекты соответствующие этим организационным единицам, а эти объекты специфицируются классами. В очень больших системах проект может содержать несколько экземпляров для каждого из этих уровней. Для небольших вы можете не использовать некоторые из уровней абстракции для организации системы.

Итак, как же все таки правильно разбить систему на подсистемы? Как распределить функциональность всей системы между подсистемами? Как должны взаимодействовать подсистемы? В результате многолетней практики я выработал некоторые рекомендаций, которые оказались полезными при определении подсистем.

- Подсистема должна быть максимально автономна от других подсистем
- Подсистемы должны обеспечивать инкапсуляцию путем использования портов и операционных контрактов.

- Внутренние элементы одной подсистемы должны быть более тесно связаны друг с другом, чем с элементами других подсистем (часто для взаимодействия между ними не будут использовать порты).
- Подсистема должна содержать элементы, которые совместно реализуют небольшой набор связанных друг с другом функций системы.
- Подсистема должна иметь подробно специфицированный набор интерфейсов, в который должны быть включены:
    - Набор сервисов, предоставляемых подсистемой
    - Набор сервисов, которые требуются от других подсистем
    - Пред- и пост- условия для каждого из сервисов
    - Ограничения и инварианты для каждого из сервисов
- Подсистема должна использовать общую аппаратуру для реализации ее функциональности.
- В многопроцессорной системе, как правило, подсистема полностью выполняется на одном процессоре, либо содержит согласованный набор функций, выполняемых на тесно связанных друг с другом процессорах.
- Если в проекте участвуют несколько команд разработки, то разработкой одной подсистемы должна заниматься одна команда.

Разумеется, эти рекомендации не являются непреложными правилами: создание хорошего проекта – все также остается как инженерной дисциплиной, так и искусством.

Порты часто используются при разбиении системы на подсистемы. Порты – это паттерн проектирования, который:

- Позволяет явным образом делегировать сервисы, определенные для подсистемы, внутренним элементам этой подсистемы.
- Способствует инкапсуляции, так как не позволяют внешним клиентам знать о внутренностях структурного класса.
- Явно специфицируют контракты интерфейса.
- Привносят дополнительный уровень сложности и добавляют накладные расходы по производительности и использованию памяти, которые не всегда могут быть полностью исключены.

Из-за накладных расходов при использовании портов, их не рекомендуется использовать повсеместно; однако для подсистем и архитектурных объектов порты предоставляют преимущества, которые перевешивают накладные расходы по их использованию.

Для данного задания, пользуясь вышеописанными рекомендациями, определите подсистемы для системы управления дорожным перекрестком Роадраннер и для системы БЛА Койот. Это означает, что необходимо определить объекты-подсистемы (или классы, если вам больше нравится) и отобразить подсистемы для системы на структурной диаграмме, на которой самый внешний класс будет являться системой в целом, а внутренние части этого класса будут представлять подсистемы. Добавьте порты и соедините подсистемы между собой, как вы считаете нужным. Учитывайте, что эти порты и связи могут измениться по мере того как мы будем выполнять дальнейший анализ и проектирование.

Модель для Роадраннер содержит пакет "_Система\Архитектура системы" [2]. Поместите классы подсистем, объекты и диаграммы в этот пакет. Помните о том, что система БЛА Койот может содержать несколько уровней подсистем (таких как, воздушное судно, наземная станция, а также подсистемы, входящие в эти основные системы). Поскольку для БЛА Койот используются несколько моделей, определите подсистемы в системной модели в пакете "Архитектура".

Задание 4.3 Привязка операционных контрактов к подсистемам

Как мы уже видели с вами, даже в проектах по разработке на основе моделей требования фиксируются различными способами. Одним из основных способов фиксации требований являются *сценарии*, которые содержат упорядоченную последовательность задействования сервисов при взаимодействии с системой. Сценарий – это определенная последовательность действий в варианте использования для получения конкретного результата, в котором в определенном порядке задействуется определенный набор сервисов. На самом верхнем уровне абстракции ("черный ящик") вся система изображается с помощью одной линии жизни на диаграмме последовательностей. Однако, элементы модели уровня системы сами выполняют очень мало работы; их роль сводится в основном к организации и управлению поведением своих внутренних частей с целью достижения операционных целей системы. Это означает, что мерой для оценки "качества" разбиения системы на подсистемы является способность подсистем взаимодействовать друг с другом для достижения операционных целей системы. Если они могут это сделать эффективно, то разбиение системы на подсистемы считается выполненным "хорошо", в противном случае говорят, что оно "нуждается в улучшении". Набор сервисов, которые задействуются в наборе сценариев для двух взаимодействующих элементов, находящихся на одном уровне абстракции, составляют "операционные контракты" этих элементов и будут использоваться для спецификации интерфейсов портов между этими элементами.

В этом задании вам предлагается рассмотреть по одному варианту использования для каждой из систем и по одному или несколько сценариев. Вам нужно взять сценарий и операционные контракты, которые он определяет, и привязать их к подсистемам, показав, каким образом подсистемы взаимодействуют друг с другом для реализации поведения, определяемого вариантом использования.

На Рис. 4-2 представлены варианты использования для системы управления дорожным перекрестком. В этом задании необходимо выбрать один вариант использования и один или несколько сценариев, и определить для них представления с детализацией уровня подсистем ("белого ящика"), определив роли, которые играют подсистемы в реализации сценариев.

[2] Мне нравится использовать подчеркивание в пакете _Система, чтобы гарантировать, что он будет первым в алфавитном порядке, даже несмотря на то, что в более поздних версиях Rhapsody это не обязательно.

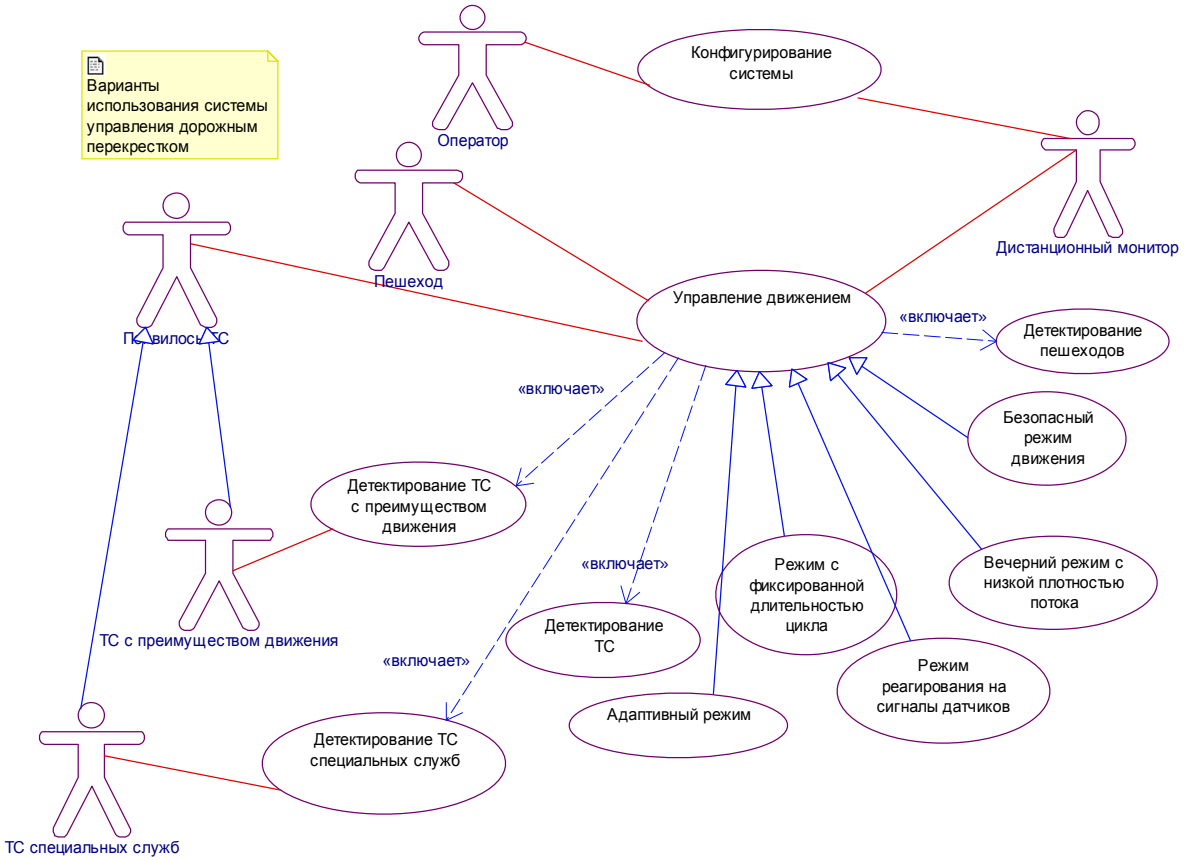

Рис. 4-2 Варианты использования системы Роадраннер

Мы рассмотрим вариант использования "Режим реагирования на сигналы датчиков". Вспомните, что в этом режиме система получает и реагирует на сигналы от транспортных средств и пешеходов, соответствующим образом включая сигналы светофоров. На Рис. 4-3 показано как данный сценарий отрабатывает в определенной ситуации.

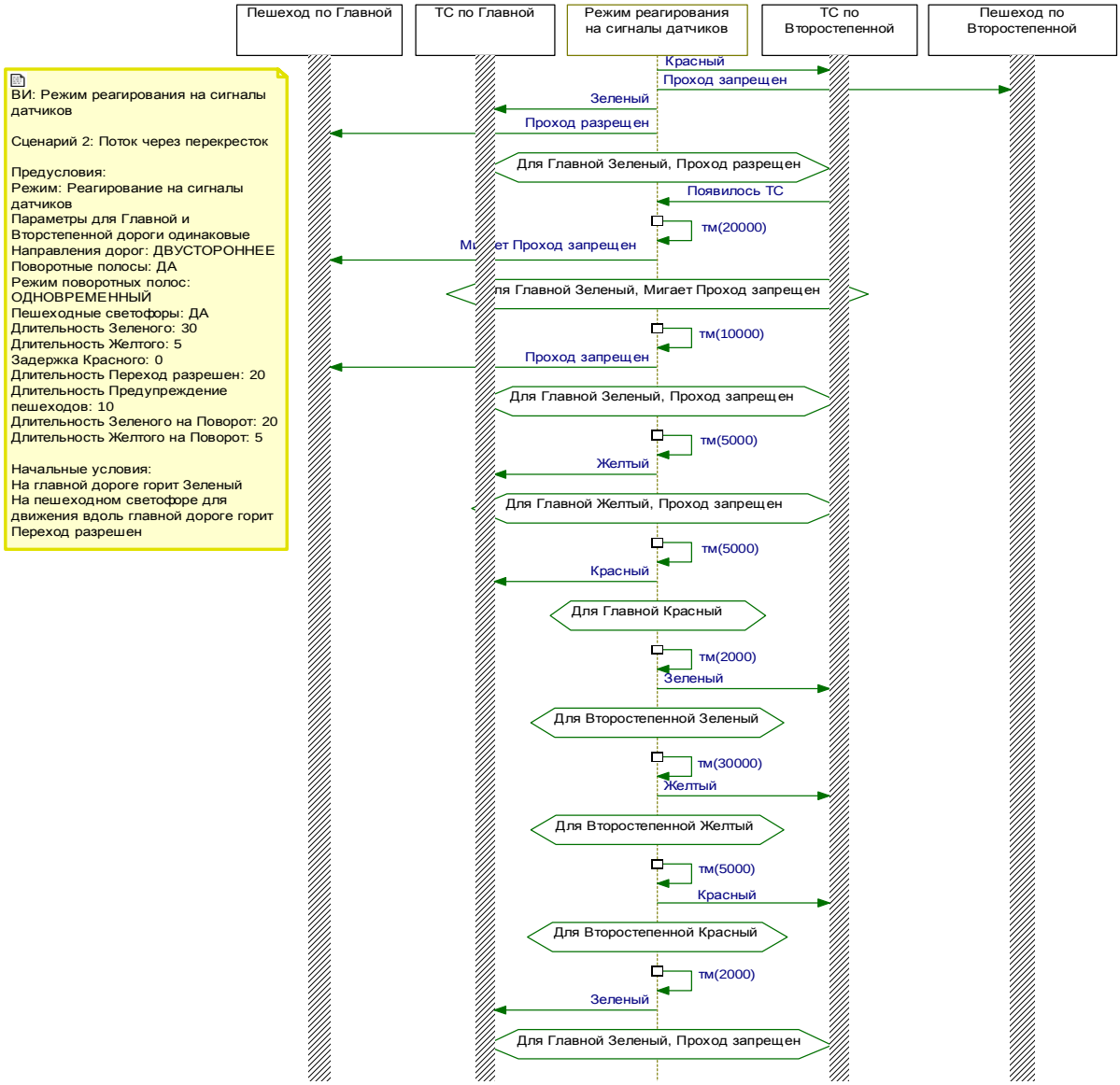

Рис. 4-3: Сценарий ВИ "Режим реагирования на сигналы датчиков" для Роадраннер

На следующем рисунке показан другой сценарий для того же самого варианта использования. Чтобы проиллюстрировать различные возможные способы моделирования, в предыдущем сценарии (Рис. 4-3) изображена линия жизни для варианта использования системы; на следующем рисунке (Рис. 4-4) изображена линия жизни объекта Система. Между первым и вторым способом нет существенных различий, и вы можете выбрать любой из них. Если вы изображаете линию жизни варианта использования, то он представляет собой систему в данном варианте использования; если же вы изображаете линию жизни объекта Система, она все равно представляет систему; но в этом случае вы все равно фокусируетесь на функциональности в данном варианте использования. Несмотря на использование двух различных способов, эти два сценария иллюстрируют различные реализующиеся пути в варианте использования "Режим реагирования на сигналы датчиков". В первом случае, транспортное средство приближается по второстепенной дороге, заставляя контроллер перекрестка переключать сигналы светофора. Во втором случае, пешеход или группа пешеходов подает сигнал о том,

что он (они) хотят перейти улицу, что инициирует переключение сигналов светофора, позволяющее пешеходам перейти улицу.

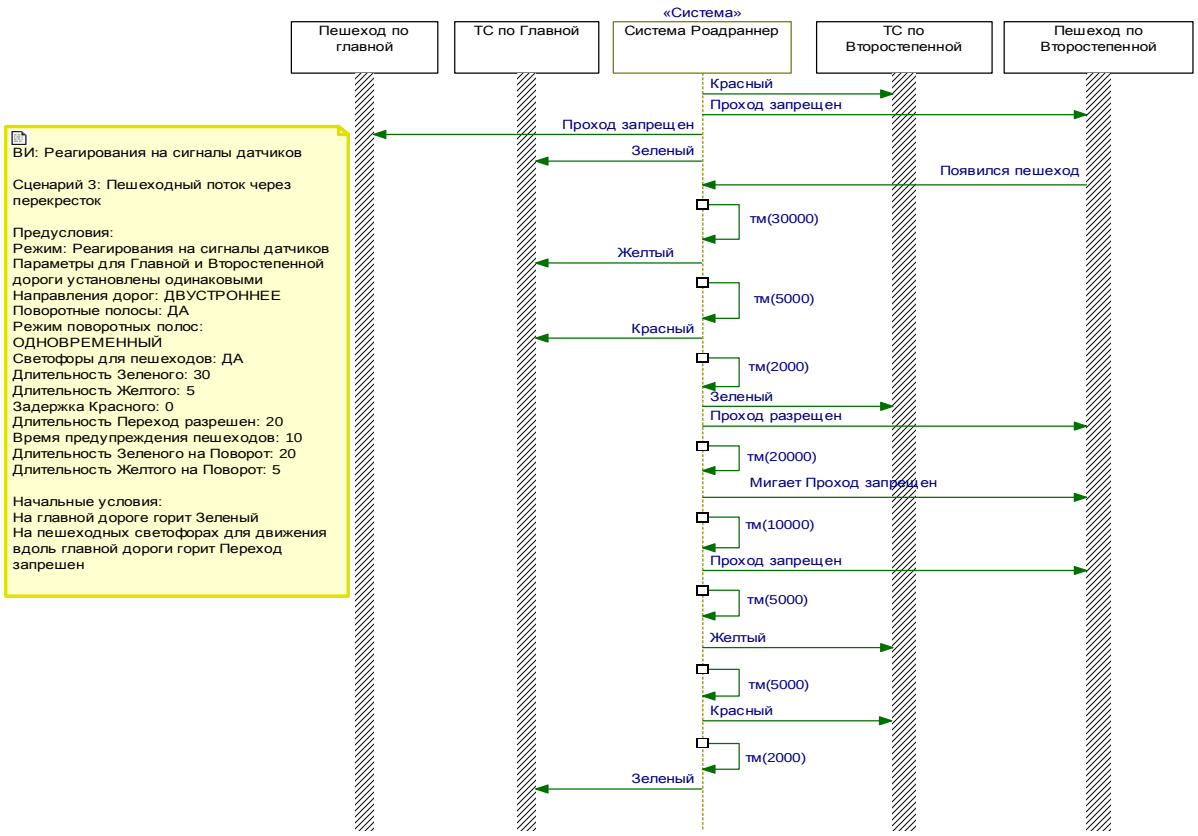

Рис. 4-4 Сценарий 3 для Роадраннер

Рассмотрите эти два сценария и проработайте их, включив в них подсистемы, определенные для системы управления дорожным перекрестком Роадраннер. Покажите каким образом эти подсистемы взаимодействуют друг с другом для реализации требуемого поведения уровня системы.

Во второй части этого задания вам необходимо сделать то же самое для системы БЛА Койот. Мы хотим привязать операционные контракты, определенные для варианта использования "Поиск по территории". Вариант использования и его описание показаны на Рис. 4-5. Разумеется, система БЛАК – это достаточно большая и сложная система. Как показано на Рис. 4-6, вариант использования "Поиск по территории" включает в себя другие варианты использования. Поскольку "Поиск по территории" -- это разновидность варианта использования "Выполнение миссии", он неявным образом включает в себя навигацию и управление полетом БЛА, управление каналом связи, получение и обработку данных наблюдения и т.д. Поэтому когда мы моделируем операционные контракты для варианта использования "Поиск по территории" задействуется большая часть поведения системы БЛАК. Мы несколько упростим здесь поведение системы, чтобы не усложнить задание чрезмерно, но мы включим достаточный объем, чтобы вы смогли получить полное представление. Не очень пугайтесь сложностью системы – в конце концов, на практике разработка требований и системный анализ для летательного аппарата занимает несколько лет даже у команды высококвалифицированных, специализирующихся в данной области, инженеров.

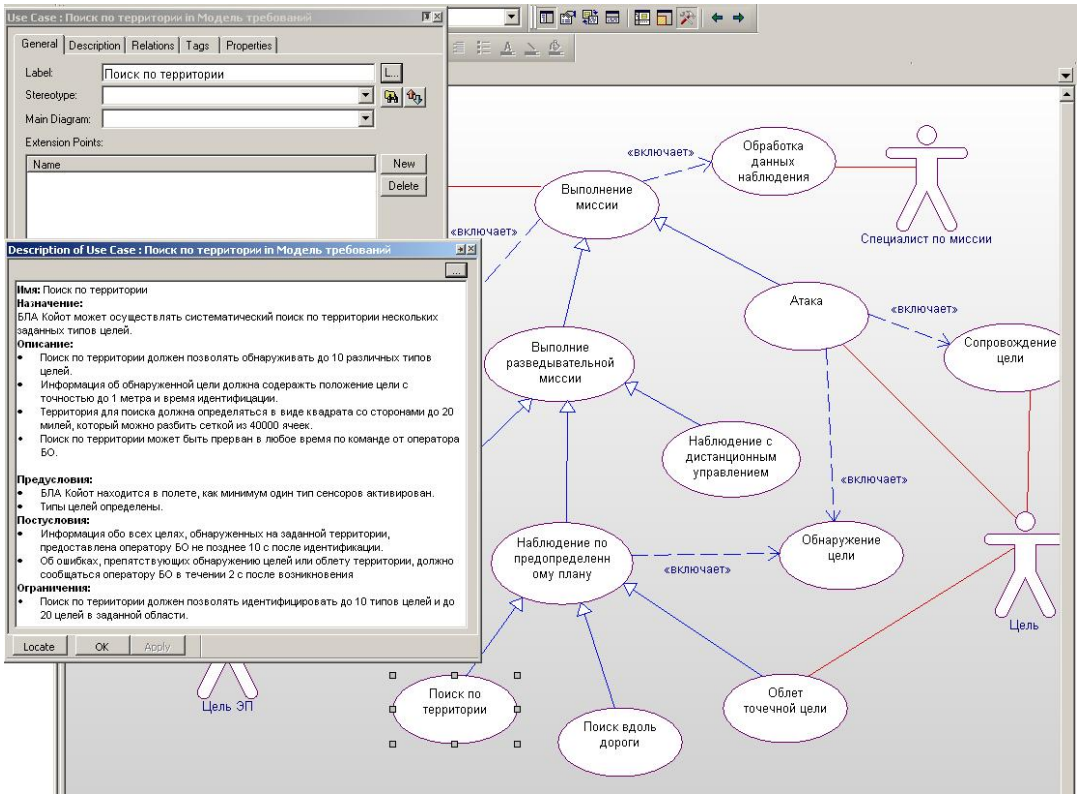

Рис. 4-5 ВИ "Поиск по территории" для БЛА Койот

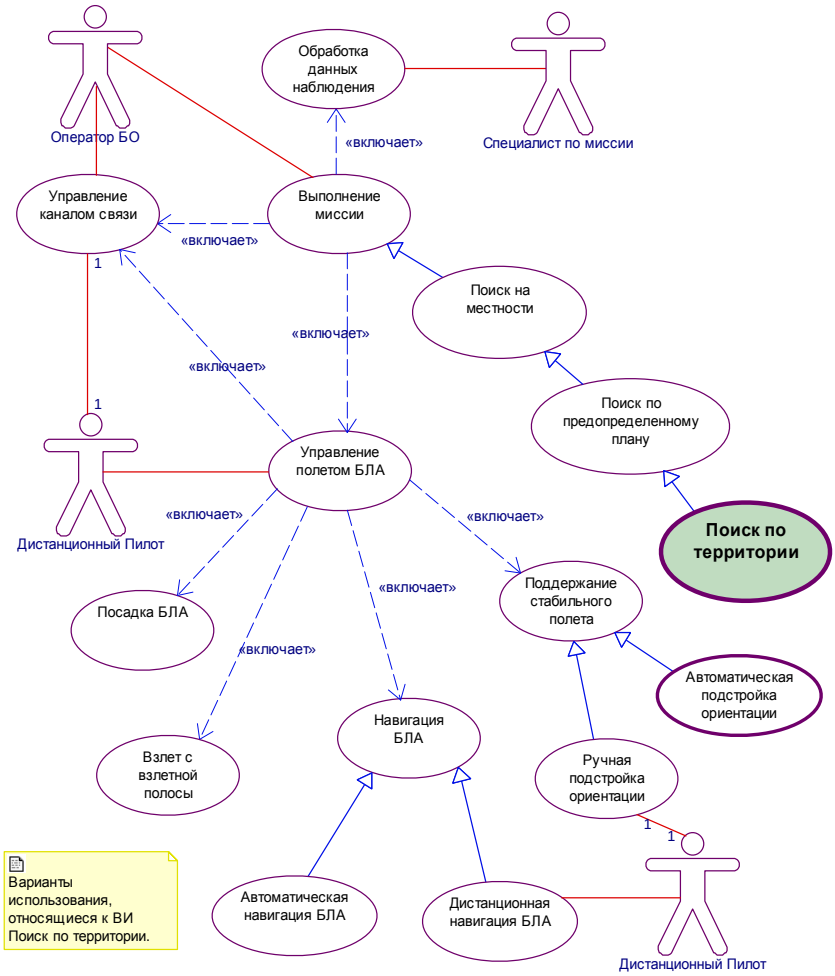

Рис. 4-6 Варианты использования, связанные с ВИ "Поиск по территории"

Из-за значительной длины сценария (а также для иллюстрации техники), сценарий разбит на несколько диаграмм последовательностей. Первая из них – в определенном смысле "главное", или высокоуровневое представление системы. Она разбивается на три под-диаграммы, отображаемые с помощью ссылок на главной диаграмме. Каждая из них ссылается на отдельную диаграмму, фокусирующуюся на определенной части всего сценария. На самом деле, декомпозиция этой высокоуровневой диаграммы последовательности выполняется аналогично декомпозици вариантов использования, и это не случайно.

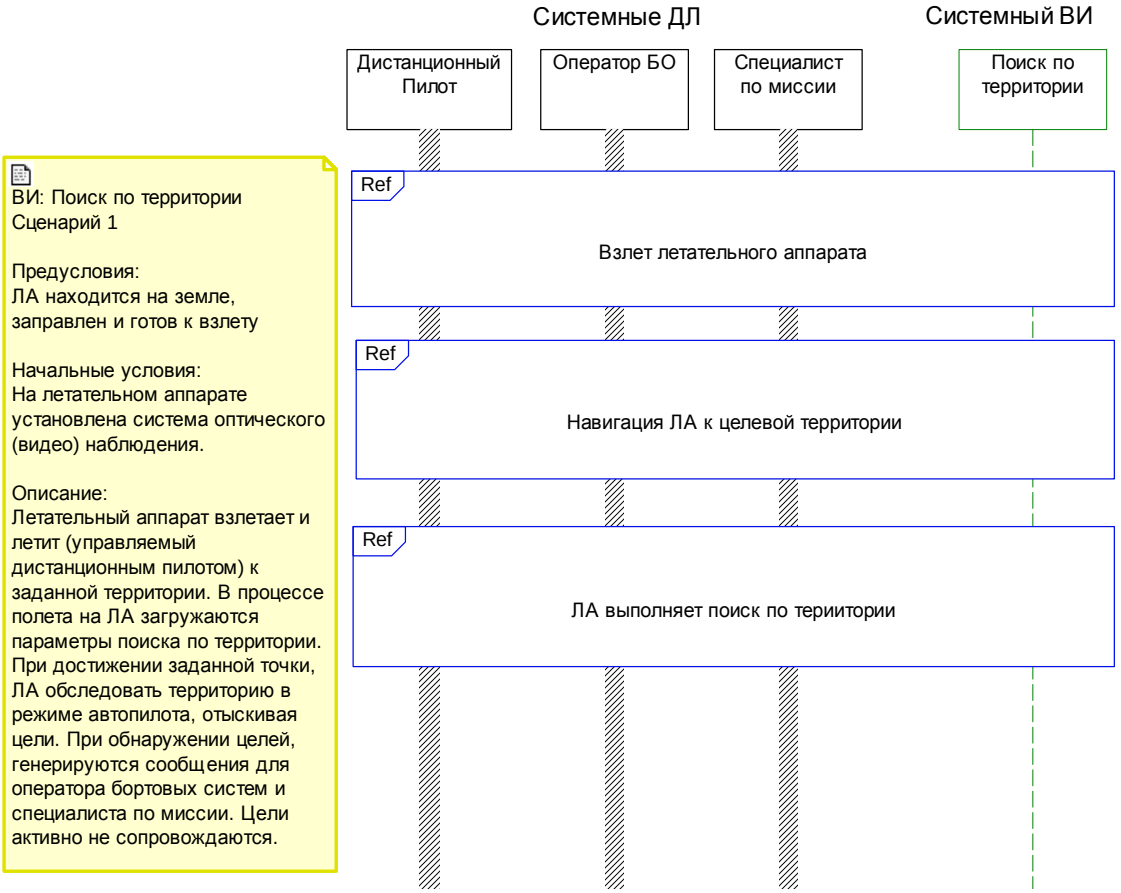

Рис. 4-7 Сценарий 1 для ВИ "Поиск по территории" - высокоуровневое представление

На Рис. 4-8 показана проработка сценария взлета летательного аппарата. Заметим, что отображенный сценарий упрощен и используется только для отображения *набора* команд от пилота летательному аппарату и набора ответных сигналов и сообщений о состоянии от летательного аппарата пилоту, но не их точную последовательность. Это демонстрирует полезный способ для моделирования такого типа систем - которые непрерывно управляются пилотом, но управление реализуется набором дискретных команд и сообщений. Заметьте использование оператора *пар* ("параллельный") - в данном случае он обозначает независимость этих двух наборов взаимодействий и возможность их произвольного перекрытия. В верхней части представлены команды, поступающие от пилота, а в нижней – сообщения о состоянии, которые отправляет летательный аппарат.

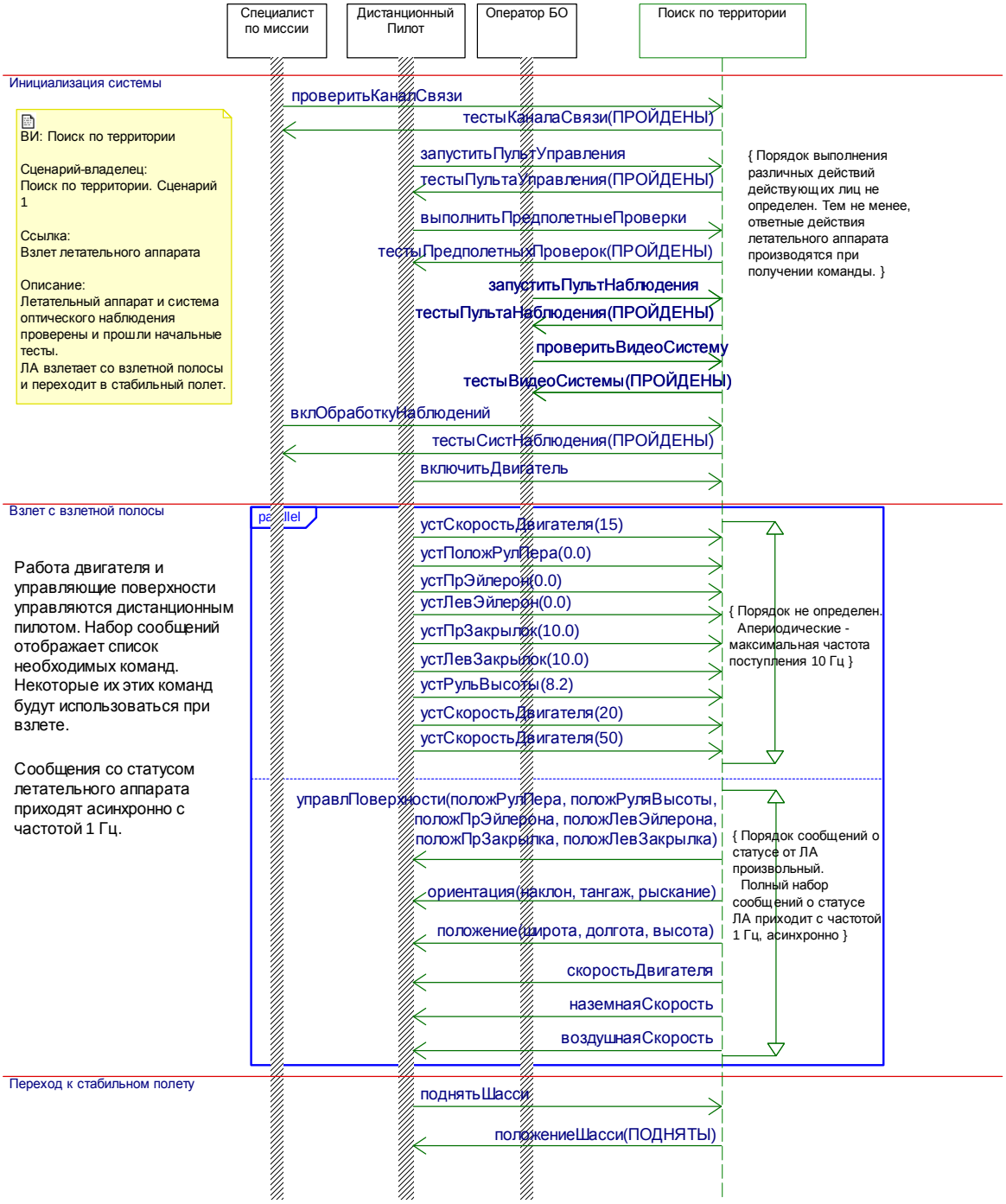

Рис. 4-8 Сценарий 1 для ВИ "Поиск по территории" - Взлет летательного аппарата

После взлета летательного аппарата он должен направляться к интересующей точке на местности . Разумеется, удаленный пилот в данном случае управляет летательным аппаратом дистанционно. Было бы неправильно фиксировать здесь сотни или даже тысячи возможных вариантов последовательностей сообщений и команд, реализуемых в процессе подстройки полета летательного аппарат по командам дистанционного управления. Как на предыдущей диаграмме последовательностей, мы ограничились определением набора команд которые *могут* быть отданы (см. Рис. 4-9). Мы добавили ограничение, чтобы показать, что отдаваемые пилотом команды могут следовать в произвольном порядке – это означает, что пилот может отдавать любые команды в любое время и в любом порядке, но частота следования команд не должна превышать 10 команд за 1 секунду.

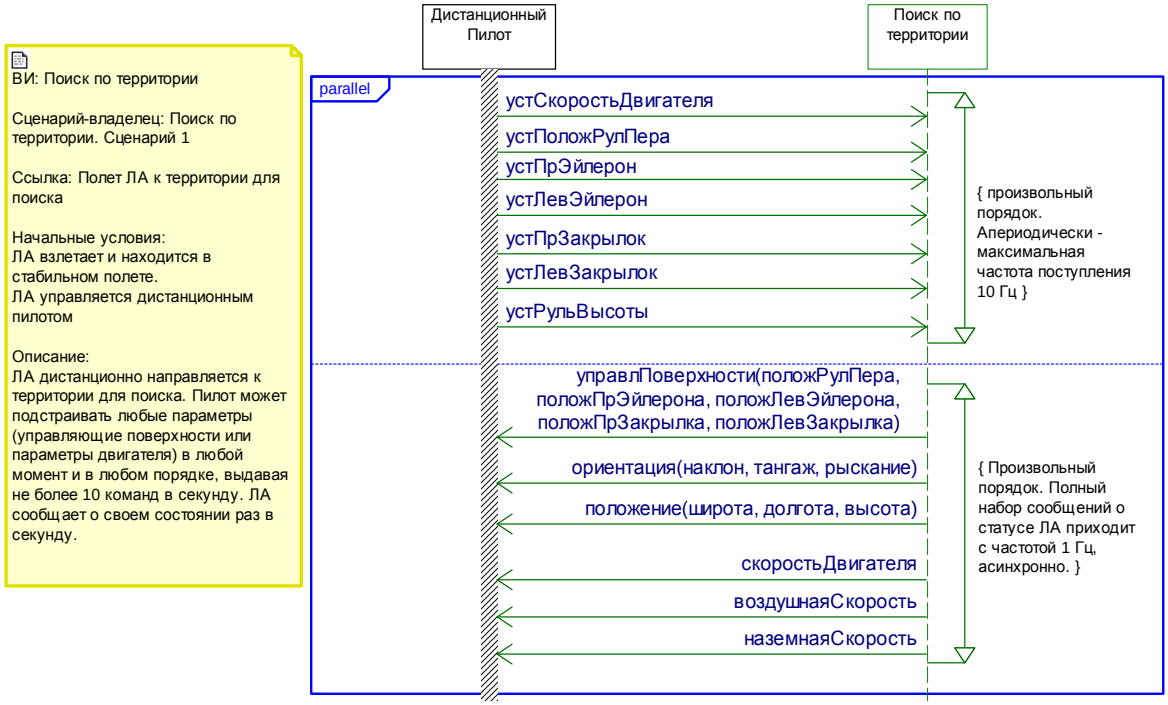

Рис. 4-9 Сценарий 1 для ВИ "Поиск по территории" - Дистанционная навигация

На следующем рисунке, Рис. 4-10, воздушному судну отдается команда выполнять поиск по заданной территории. В рамках этой задачи, летательный аппарат определяет сетку для поиска по прямоугольной области, и облетает эту область последовательно колонка за колонкой, выполняя поиск целей. При обнаружении цели, ее положение записывается и передается наземной станции. Летательный аппарат также поддерживает список обнаруженных целей. Выполняя поиск по территории, летательный аппарат управляется внутренним автопилотом.

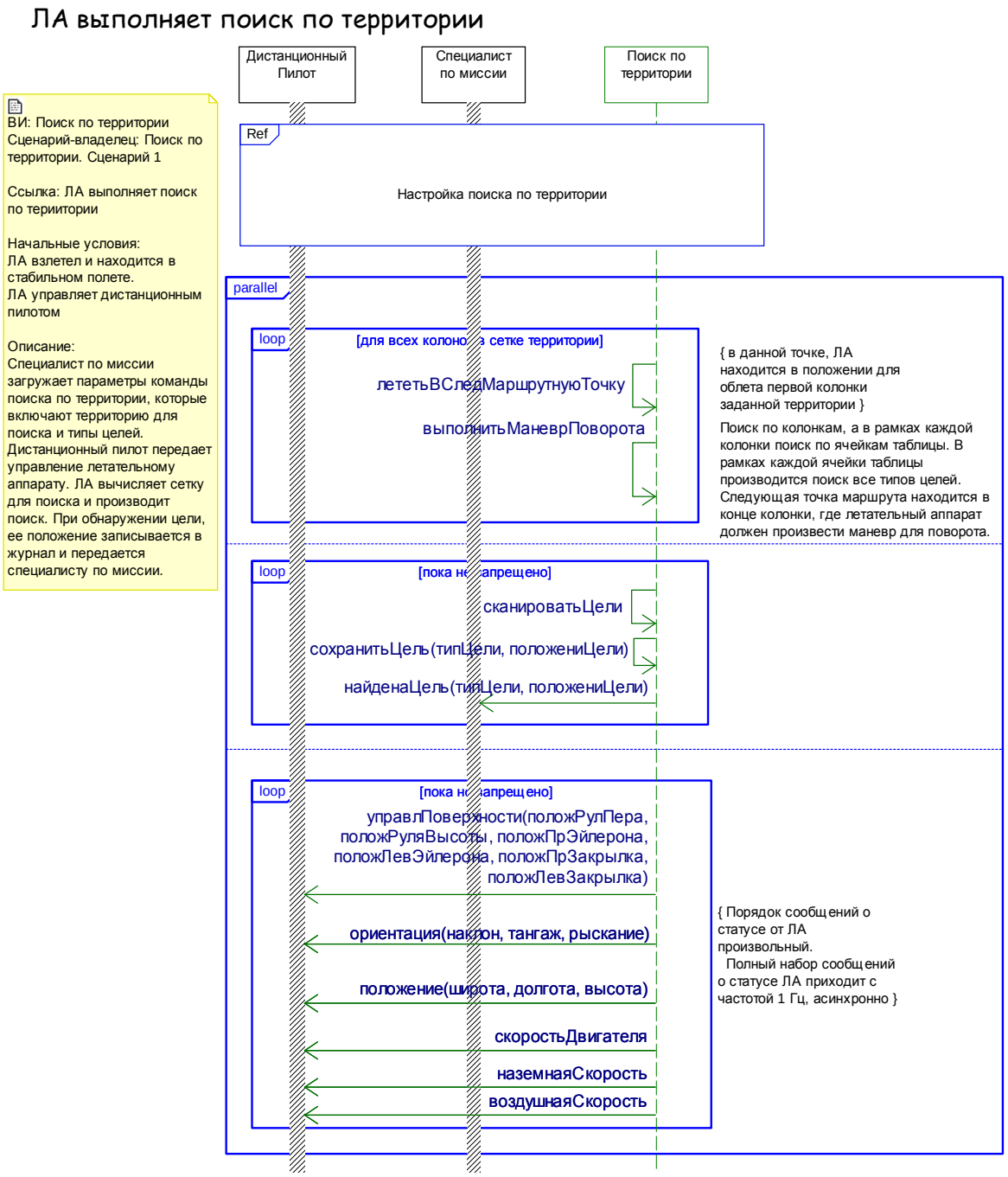

Рис. 4-10 Сценарий 1 для ВИ "Поиск по территории" - Летательный аппарат выполняет поиск по территории

Вложенный сценарий, изображенный на Рис. 4-10, также содержит вложенный сценарий для определения области поиска. Он показан на Рис. 4-11.

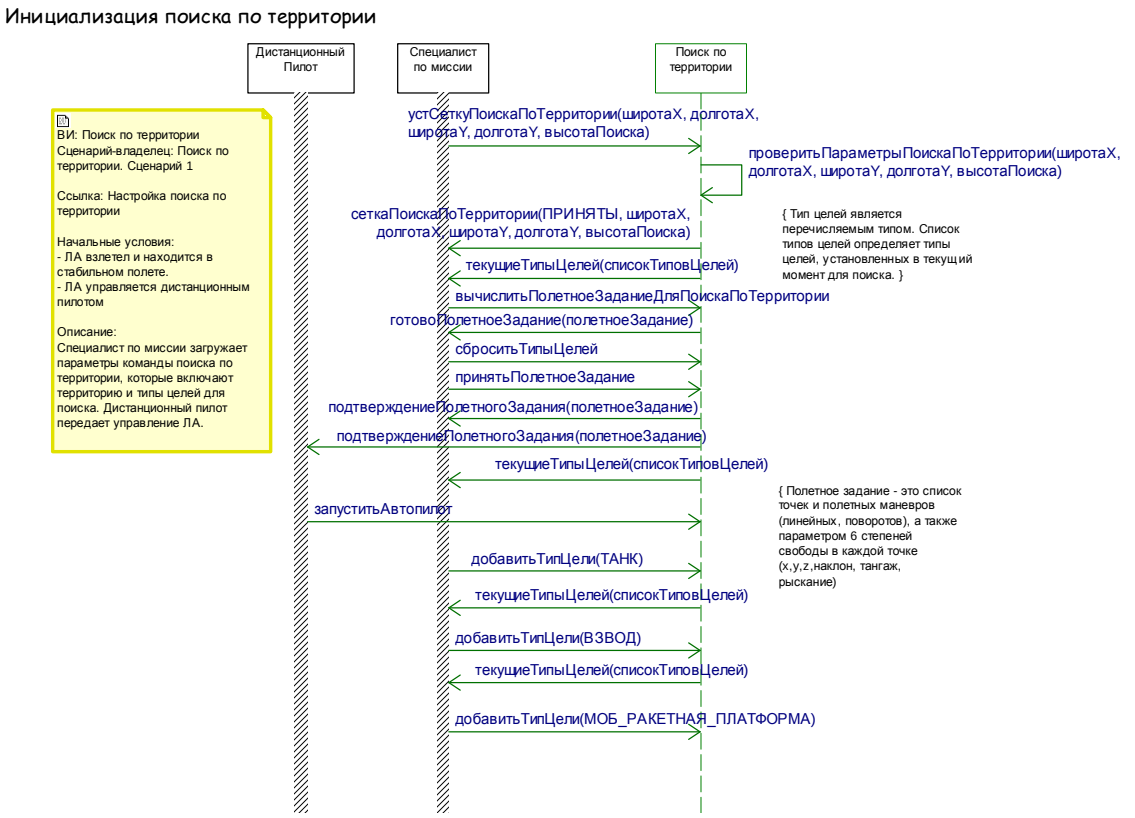

Рис. 4-11 Определение области поиска

В качестве второй части этого задания, вам необходимо привязать операционные контракты, определенные для варианта использования системы БЛА Койот "Поиск по территории", к определенным подсистемам. Не забывайте, что система БЛА Койот – это большая система. Здесь мы рассматриваем только один сценарий для одного варианта использования. При реальной разработке системы для одного варианта использования может быть определено десять и более различных сценариев; и эту операцию вам необходимо проделывать для каждого сценария всех вариантов использования. После того как будут определены основные сценарии для варианта использования, дополнительные сценарии "хорошей погоды" будут добавлять к вашей модели очень небольшое количество новых операционных контрактов. Сценарии "плохой погоды", моделирующие обработку исключительных ситуаций, будут добавлять большее количество дополнительных сервисов и поведения. В любом случае, это полезно проделать, чтобы убедиться, что все необходимые сервисы предоставляются каким-либо элементом системы, общее поведение системы согласовано, а системная архитектура удовлетворяет всем требованиям к системе.

Задание 4.4 Определение вариантов использования для подсистем

Как правило, большие проекты выполняются несколькими командами. Самый распространенный способ организации работы нескольких команд – это назначить отдельную команду для работы над отдельной подсистемой. Подсистема – это наиболее крупная архитектурная часть физической системы. Она может состоять только из программных компонентов или включать в себя компоненты, относящиеся к различным инженерным дисциплинам – обычно, программному обеспечению,

электронике, механике и химии. Независимо от того располагаются ли эти команды на одной или разных территориях, для успешной работы им необходима два типа информации.

Во-первых, им необходимо понимать требования, которые относятся к их подсистемам. Например, если я работаю в команде, разрабатывающей энергетическую подсистему для космического корабля, мне необходимо понимать какие возможности от нее ожидают ее пользователи – действующие лица системы и окружающие подсистемы. Мы организуем системные требования по вариантам использования; точно также стоит сделать и для подсистем.

Во-вторых, мне необходимо знание об операционном контексте в котором должна работать система - кто пользователи сервисов нашей подсистемы и кто предоставляет сервисы для нашей подсистемы. Мне также необходимо знать какие интерфейсы предоставляет моя подсистема этим пользователям, включая пред- и пост- условия, состав и типы используемых данных, ограничения, накладываемые требованиями к качеству, а также допустимые последовательности использования этих сервисов. Такую же информацию необходима нам по интерфейсам, используемых нашей подсистемой.

Эти два типа информации содержатся в двух источниках - модели требований к системе и в системной архитектуре. Эти два типа информации являются взаимозависимыми. Выбрав другую структуру для архитектуры системы, мы получим другой набор вариантов использования для подсистем.

Итак, мы имеем вопрос: "Как на практике определять варианты использования для подсистем?". Как было только что сказано, в для определения вариантов использования подсистем необходимы два источника информации: модель требований и системная архитектура. На основе них необходимо выполнить *декомпозицию* вариантов использования системы на варианты использования подсистем, каждый из которых реализуется отдельной подсистемой. С использованием языка UML это будет производиться с помощью зависимостей со стереотипом <<включает>>. Обычный сценарий для определения вариантов использования подсистем таков:

Для каждого варианта использования системы:

- Выделите группы требований, которым должна удовлетворять отдельная подсистема;
- Создайте вариант использования уровня подсистемы, для организации данной группы требований;
- Добавьте зависимость со стереотипом <<включает>>, чтобы связать вариант использования уровня подсистемы с родительским вариантом использования уровня системы.
- Привяжите вариант использования к подсистеме.
- Переместите вариант использования подсистемы в область модели, где располагаются требования к данной подсистеме.

В результате для каждой из подсистем у вас должен получиться определенный набор вариантов использования, в которых зафиксированы требования к этой подсистеме.

Каждое операционное требование и требование к качеству операций должно быть представлено в варианте использования как минимум одной подсистемы; некоторые требования будут представлены во всех или во многих подсистемах. Некоторые требования должны быть декомпозированы на производные требования, каждое из которых будет привязано к одному варианту использования для подсистемы. Основная идея состоит в том, что модель требований для подсистемы представляла полный набор требований для этой подсистемы. После этого команда по разработке подсистемы может приступать к разработке своей подсистемы более менее независимо от других команд, уверенные в том, что их подсистема без проблем встроится в общую систему и будет работать корректно. Трассировка достигается благодаря тому, что связи между вариантами использования системы и вариантами использования подсистем являются трассируемыми связями. Кроме этого, обычно добавляются ссылки на требования, содержащиеся в инструментах предоставляющих возможности по трассировке требований, таких как DOORS.

Результатом всех этих усилий являются новые диаграммы использования, относящиеся к более детальному уровню абстракции. Как правило, вы будете создавать новую диаграмму использования для декомпозиции каждого варианта использования системы. Кроме того, вы будете создавать одну или несколько новых диаграмм использования для каждой из подсистем. Эти два вида диаграмм имеют разное назначение. На диаграммах первого вида отображаются связи между одним вариантом использования системы и входящими в него вариантами использования уровня подсистем. Эти диаграммы будут размещаться в той области модели, где находятся требования к системе. На диаграммах второго вида отображается множество вариантов использования для выбранной подсистемы. Эти диаграммы будут размещаться в той области модели, которая относится к конкретной подсистеме. Варианты использования подсистемы будут размещаться в пакете подсистемы. В результате, части модели, относящиеся к отдельным подсистемам, будет переданы командам разработки подсистем и они будут прорабатывать эти модели в отдельных проектных моделях.

Выполняя первую часть этого задания, используйте подсистемы, полученные в результате определения архитектуры системы управления дорожным светофором. Возьмите варианты использования "Конфигурирование системы" и "Детектирование ТС", которые мы определили с достаточно детализацией ранее. Рис. 4-13 отображает данные варианты использования.

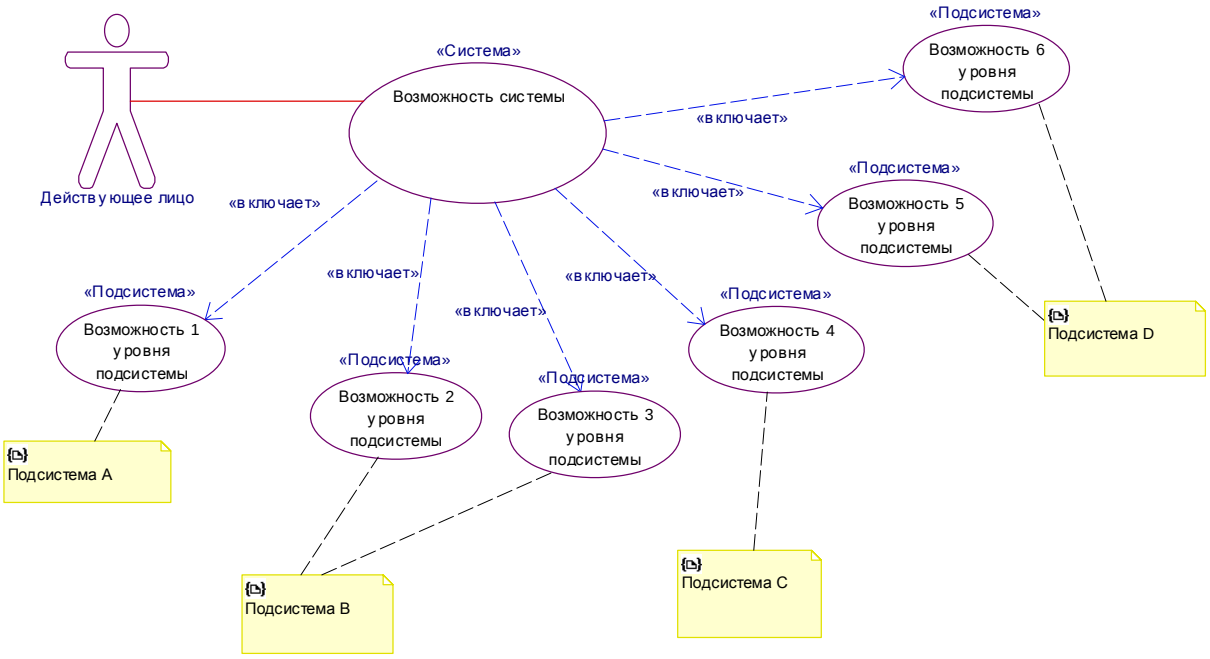

Рис. 4-12 Варианты использования подсистемы

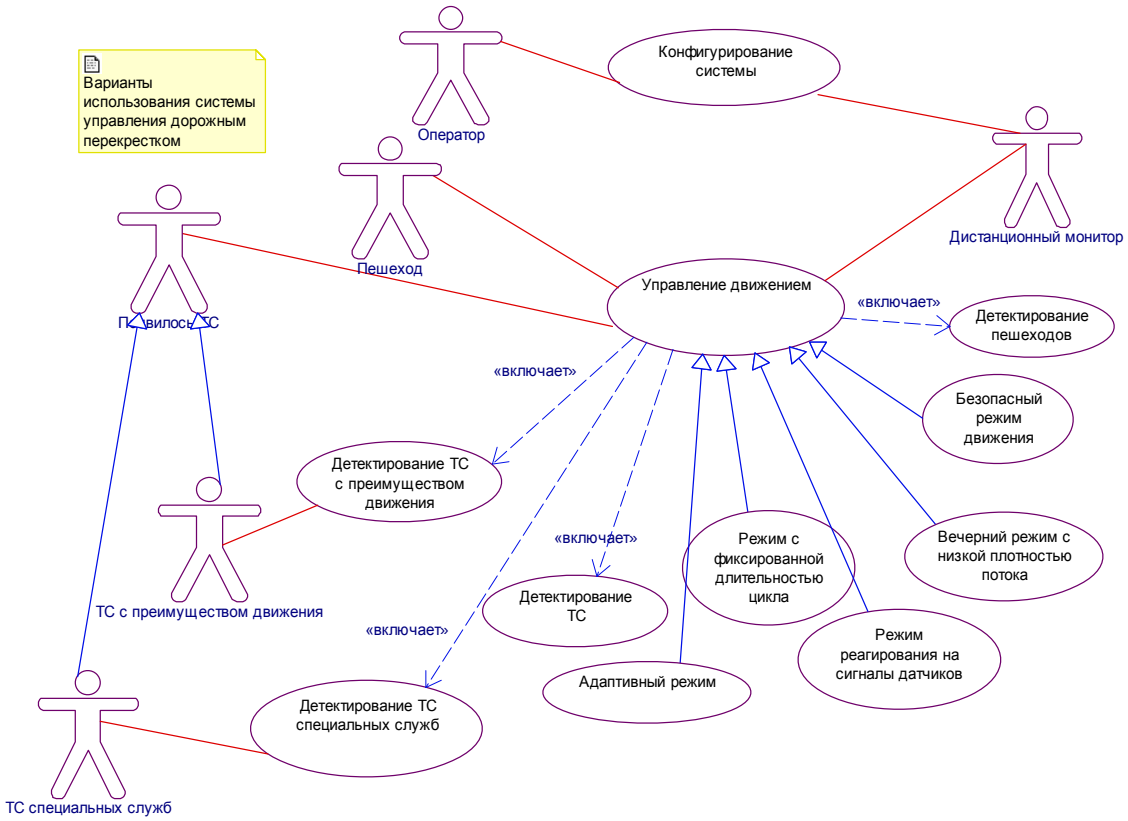

Рис. 4-13 Варианты использования Роадраннер

Архитектура для системы Роадраннер показана на Рис. 4-14. Функциональность (и требования к системе), которые представлены вариантом использования "Конфигурирование системы", необходимо связать с набором вариантов

использования подсистем. Этот вариант использования уровня системы может не накладывать никаких требований на подсистемы или может приводить к определению набора вариантов использования для различных подсистем. Очень редко встречаются случаи, когда вариант использования уровня системы будет полностью реализован только одной подсистемой.

Для завершения работы над данной частью задания, создайте новые диаграммы использования для вариантов использования "Конфигурирование системы" и "Детектирование ТС", на которых выполните декомпозицию варианта использования системы на несколько вариантов использования для подсистем. После этого создайте диаграмму использования для подсистемы "Контроллер перекрестка", на которой покажите все определенные для нее варианты использования.

Системная архитектура Роадраннер:
Назначение: Отображение входящих подсистем

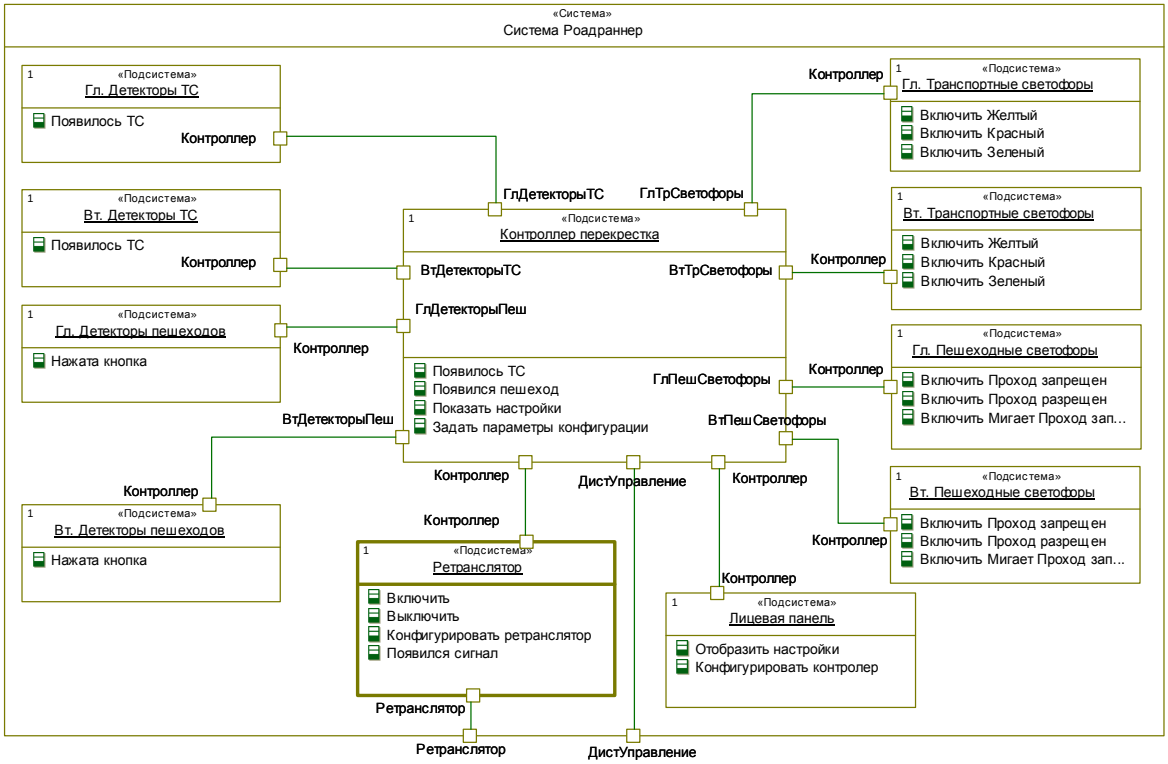

Рис. 4-14 Архитектура системы Роадраннер

Во второй части этого задания проделайте то же самое для двух вариантов использования "Ручная подстройка положения" и "Поиск по территории" системы БЛАК. Эти два варианта использования выделены на Рис. 4-15 и Рис. 4-16.

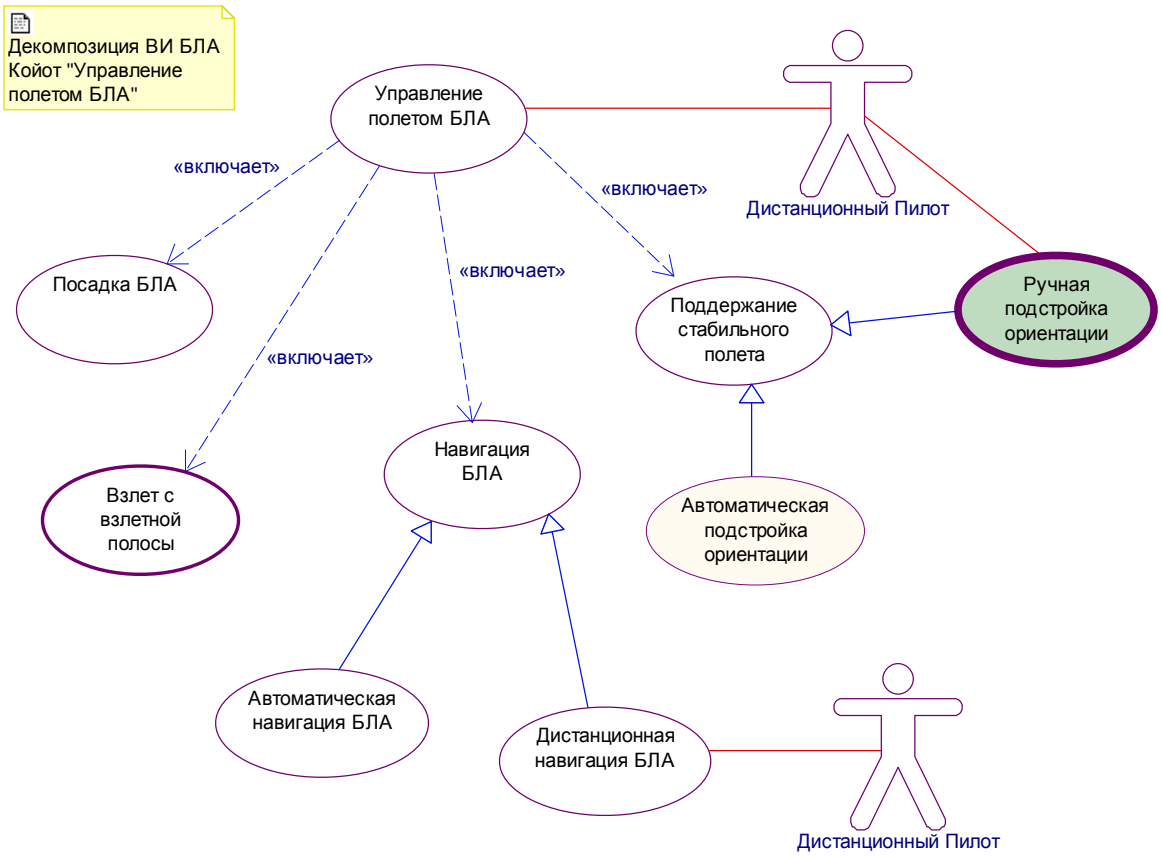

Рис. 4-15 Вариант использования "Ручная подстройка положения"

Первый из них является более простым. Возможность, предоставляемая вариантом использования "Ручная подстройка положения" означает, что пилот, управляющий летательным аппаратом с наземной станции, может контролировать положение (углы тангажа, наклона, рыскания) самолета. Для этого пилоту необходимо мониторить данную информацию, отображаемой в удобном для интерпретации виде, и иметь возможность управлять воздушным судном по каналу связи. Реализация данной возможности затронет многие подсистемы, как в составе наземной станции, так и в составе летательного аппарата. Архитектура всей системы отображена на Рис. 4-17. Две основные системы показаны на следующих двух рисунках. Структура самого летательного аппарата показана на Рис. 4-18, на котором отображены его подсистемы и связи между ними. Структура наземной станции отображена на менее сложном Рис. 4-19. Он отображает тот факт, что в системе существуют несколько ручных пультов, каждый из которых включает в себя пульт для управления летательным аппаратом и пульт для осуществления наблюдения.

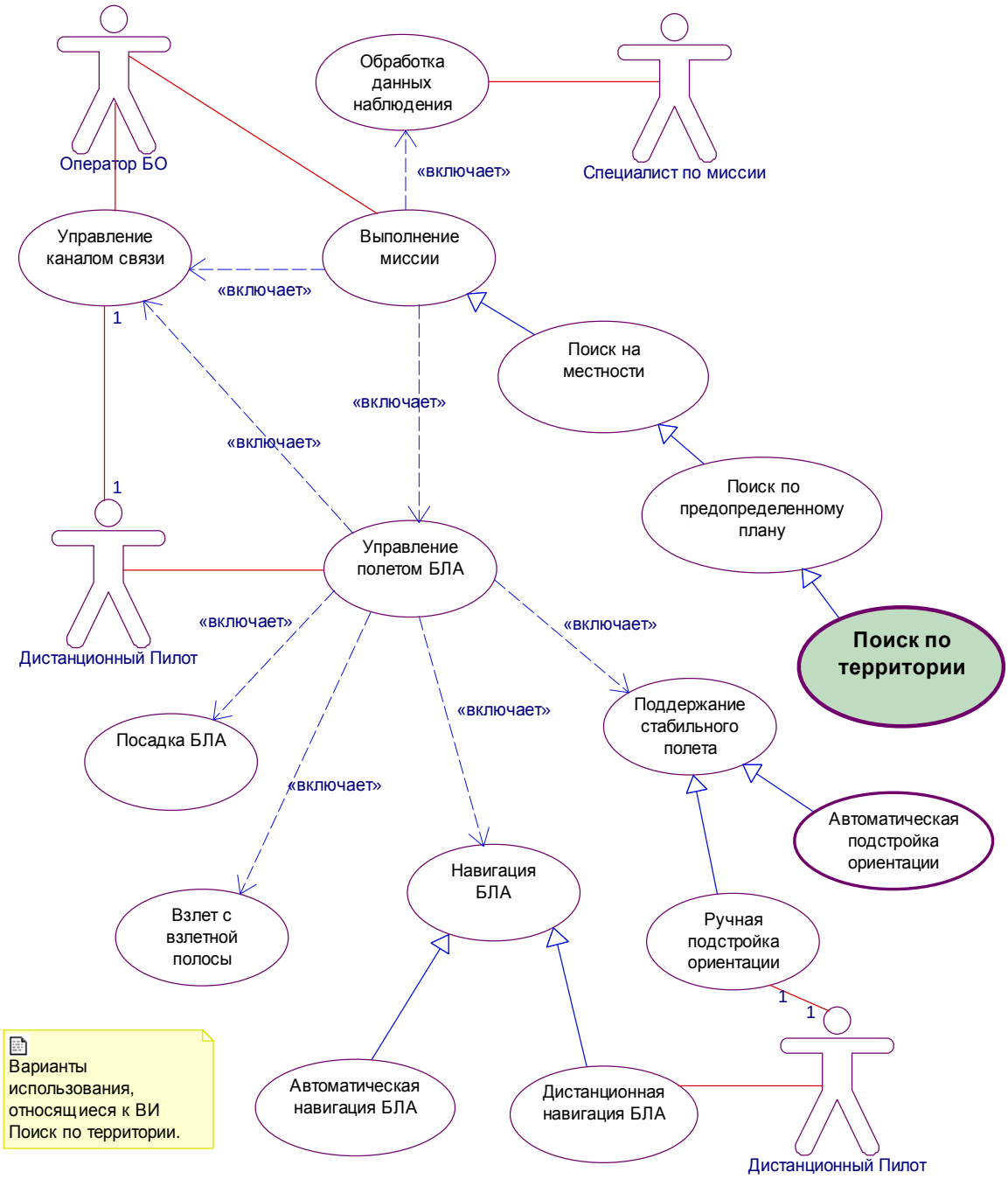

Рис. 4-16 Вариант использования "Поиск по территории"

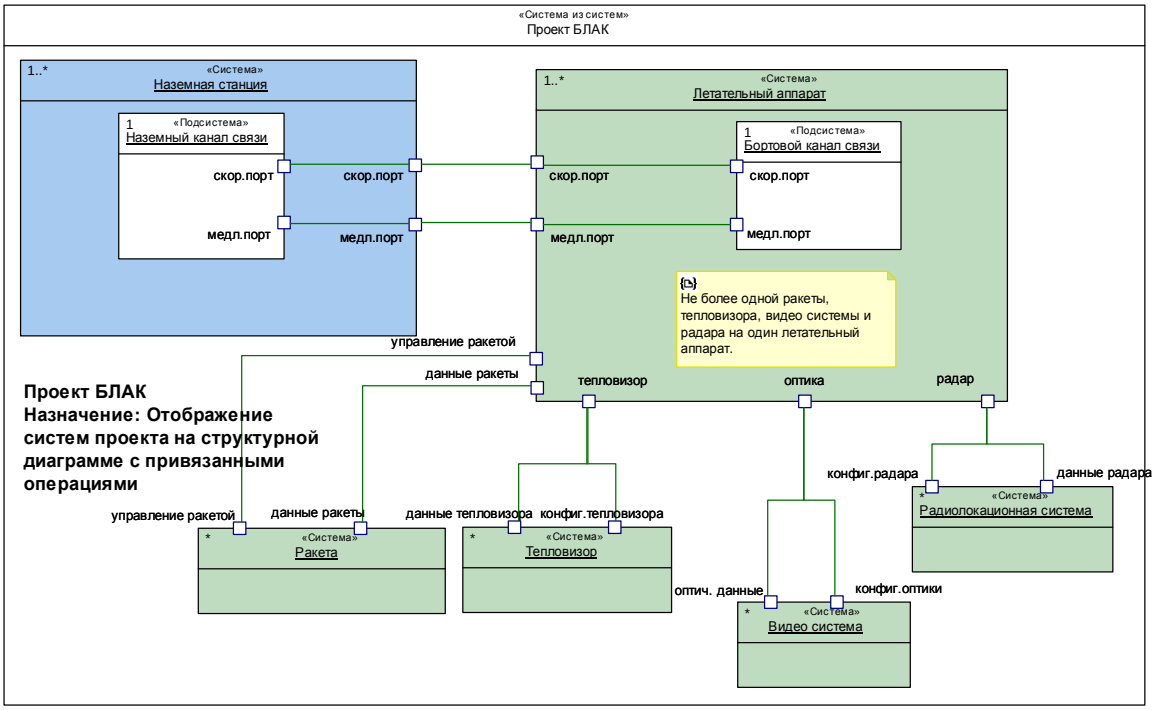

Рис. 4-17 Архитектура системы БЛАК

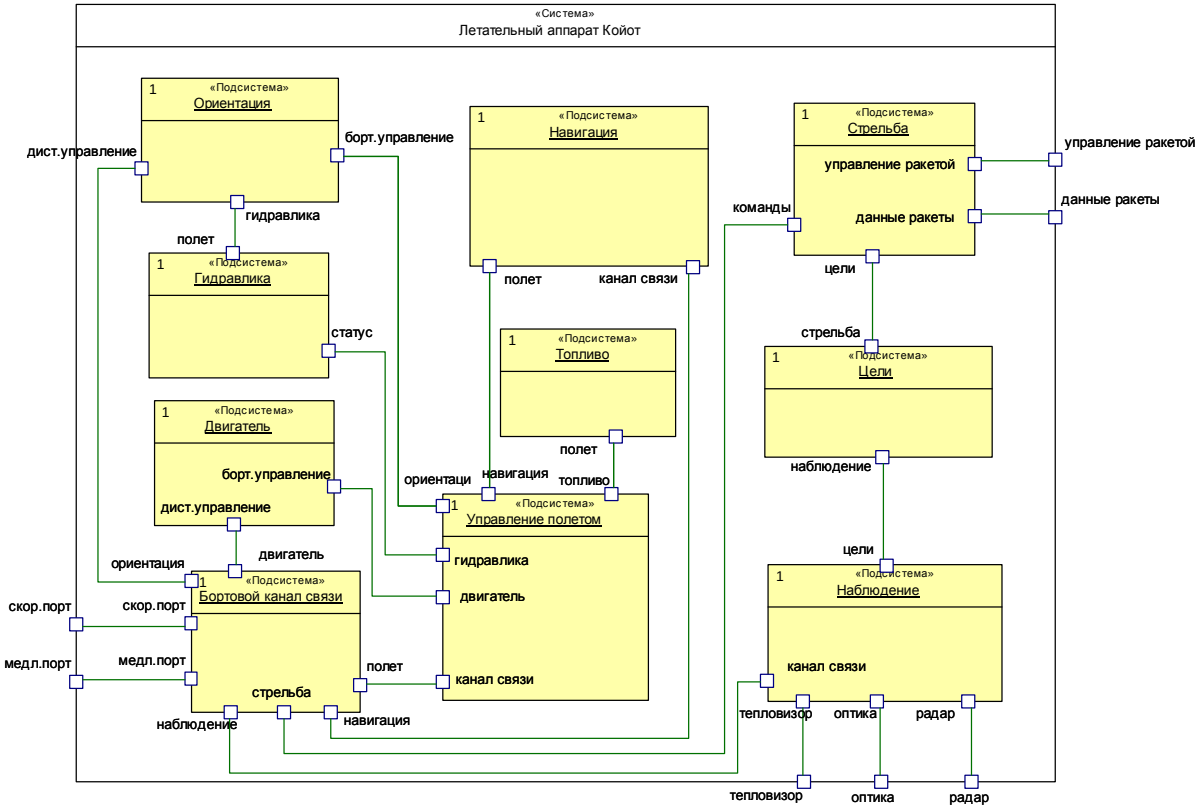

Рис. 4-18 Архитектура беспилотного летательного аппарата Койот

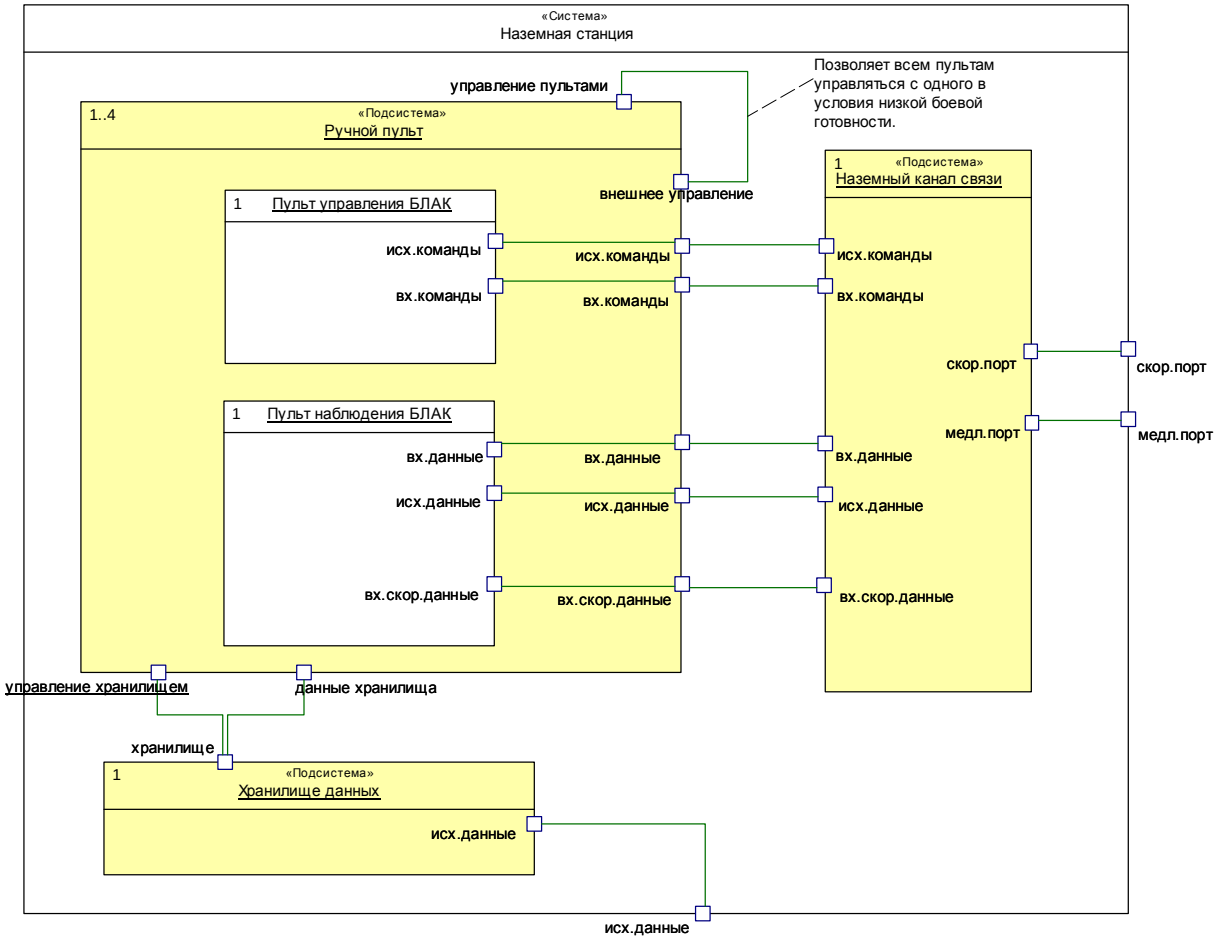

Рис. 4-19 Архитектура наземной станции системы БЛАК

Первая часть задания по идентификации вариантов использования подсистем состоит в декомпозиции варианта использования "Ручная подстройка положения" на варианты использования подсистем. Вторая часть задания состоит в декомпозиции варианта использования "Поиск по территории" на варианты использования этих же подсистем.

Далее в этой книге

Эта глава была посвящена определению системной архитектуры, в том числе спецификации подсистем и требований к ним. Следующий шаг состоит в выполнении анализа вариантов использования и создании модели, которая раньше часто называлась *существенной моделью* системы, а сейчас более часто называется *платформенно-независимой моделью*. Эта модель конструирует набор объектов, определяемых с помощью классов, отношений и элементов поведения, которые работают совместно для реализации требований рассматриваемого варианта использования. В процессе Harmony этот этап называется *объектным анализом*. Именно этому вопросу посвящена следующая глава этой книги.