Институт Транспорта и Связи

Пояснительная записка
к курсовой работе
по дисциплине «Технология конструирования программного обеспечения»
студента группы 4601
Николая Кабелева

Рига, 2000

Цель работы.

Целью данной работы является разработка программного объектно-ориентированного обеспечения для бортовой системы управления мобильным роботом в соответствии с требованиями, предъявляемыми к унифицированному процессу разработки программного обеспечения.

Предметная область.

В рамках данной работы подразумевается разработка программного обеспечения, устанавливаемого на бортовую систему управления мобильным роботом. Прежде всего, определим основные требования к роботу в целом. Как известно, *роботом* называется техническая система, осуществляющая какие-либо функции живых существ, производя при этом самостоятельное принятие решений в соответствии с заложенным в ее конструкцию алгоритмом. Любой робот состоит из блока принятия решений (обычно БЦВМ) и манипулятора(ов) (педипуляторов, движителей), возможно оборудованных схватами или другими приспособлениями для взаимодействия с рабочими телами или элементами окружающей среды. Традиционно все роботы подразделяются на три поколения:

- Роботы первого поколения – роботы, не имеющие обратной связи и осуществляющие свои функции по четко заданной программе. По сути дела роботы первого поколения являются автоматами с высокой степенью универсальности.
- Роботы второго поколения (роботы с очувствлением) – роботы, обладающие центростремительными механизмами внутренней связи, предназначенными для сбора сенсорной информации и адаптации своего поведения в соответствии с изменениями окружающей среды.
- Роботы третьего поколения (интеллектуальные роботы) – роботы, наследующие черты роботов второго поколения и добавляющие к ним развитые механизмы искусственного интеллекта, предназначенные для самостоятельной выработки алгоритмов поведения.

В рамках данной работы рассматривается мобильный робот второго поколения, способный осуществлять навигационные функции на заданном участке местности, самостоятельно отыскивая путь к установленной цели. Для выполнения поставленной задачи разрабатываемый робот должен располагать картой местности, датчиками месторасположения, каналом связи с диспетчером для выполнения аварийного останова и подсистемой внутренней проверки.
Ввиду того, что робот представляет собой набор аппаратных устройств, взаимодействующих между собой для достижения поставленной цели, в данной работы за идеологическую основу берется аппаратная архитектура робота. Такой подход

подразумевает, что в состав системы управления входит набор параллельно работающих аппаратных устройств, каждое из которых представляет собой микро-ЭВМ, располагающей своей памятью, портами ввода-вывода и микропроцессором (или набором микропроцессоров). (рис. 1). Таким образом, задача разработки

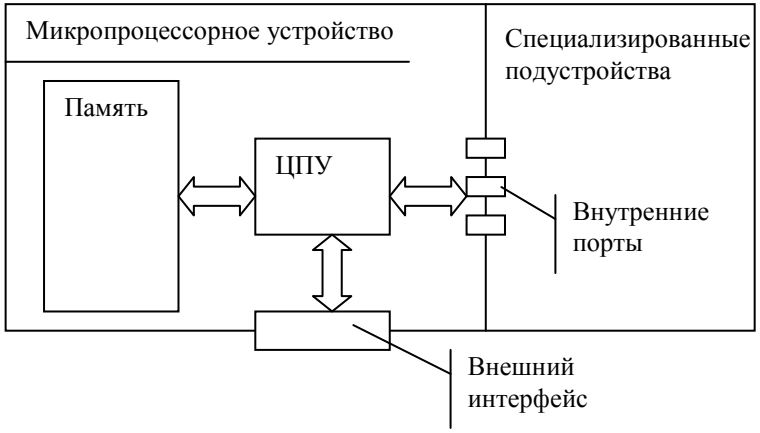

Рис. 1. Общая схема микропроцессорного устройства

программного обеспечения для системы управления мобильным роботом сводится к построению программного обеспечения для каждого из микропроцессорных устройств при соблюдении установленных дисциплин взаимодействия между устройствами. Следует отметить, что подобный подход на данный момент времени является наиболее популярным в робототехнике, так как его использование позволяет добиться следующих положительных результатов как на этапе разработки, так и производства и эксплуатации:

- Повышается модульность разрабатываемого продукта, что позволяет вести дальнейшие разработки в условиях четкой систематизированности изделия
- Повышается степень унификации как отдельных узлов, так и всего изделия в целом.
- Понижается стоимость изделия по сравнению как с вариантом использования централизованной системы управления (требуется мощный процессор, стоимость которого растет быстрее производительности), так и системе на неинтегральной базе (данное решение приемлемо только для очень простых устройств).
- Понижается стоимость работ по модификации изделия, так как в этом случае не только не требуется разработка новых аппаратных средств, но и даже их замена.

- Облегчается процесс обслуживания, локализации неисправности и ремонта изделия.

Этап Начало.

Требования к изделию в целом.

Определим набор требований, предъявляемых к разрабатываемому изделию:
Робот мобильный, оборудованный бортовой системой управления, должен выполнять следующий набор функций:

- Задание и хранение карты местности, соответствующей зоне действия робота.
- Программирование и хранение предопределенных точек, являющихся целями.
- Осуществление самостоятельного поиска и прокладка маршрута на местности к заданной целевой точке. Адаптация маршрута в режиме реального времени в случае возникновения непредвиденных обстоятельств в ходе выполнения маршрутного задания.
- Определение места расположения и сопоставление полученного результата с соответствующей точкой на карте.
- Определение состояния и проверка целостности подсистем робота с целью принятия решения о безопасности, целесообразности и возможности дальнейшего выполнения маршрутного задания.
- Наличие функции аварийного останова, выполняемого безотносительно к текущему состоянию робота.
- Наличие на борту пульта управления роботом для осуществления функций задания карты местности и программирования целевых точек, а также подачи команды на аварийный останов.
- Наличие внешнего интерфейса, позволяющего осуществлять связь по радиоканалу с диспетчерской башней, посылающей команды на выбор целевой точки, начало выполнения маршрутного задания и аварийного останова, а также производящей запросы от системы управления текущего состояния подсистем робота и текущего места расположения.

Определение внешнего окружения системы.

Исходя из поставленных требований, определим характер внешнего окружения системы и постоим набор актеров, участвующих в процессе взаимодействия с изделием (рис. 2):

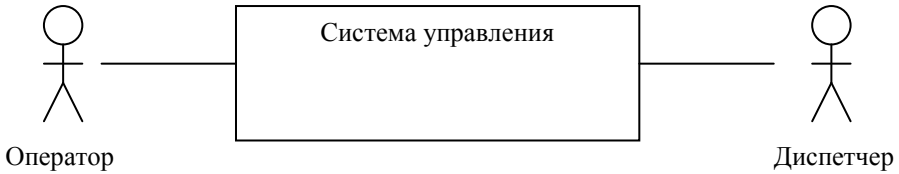

Рис. 2. Внешнее окружение системы управления

Таким образом, имеем двух актеров, взаимодействующих с системой:

- Оператор – осуществляет непосредственное задание карты местности и программирование целевых точек с расположенного на борту пульта управления.
- Диспетчер – осуществляет дистанционную подачу команд на выбор текущей целевой точки, начало выполнения маршрутного задания, аварийный останов, а также запрашивает информацию о состоянии подсистем робота и текущем месте расположения.

Идентификация Use-case'ов.

Для идентификации набора Use-case'ов определим действия, выполняемые по отношению к разрабатываемой системе каждым из актеров. Подобного рода анализ позволит нам выделить Use-case'ы первого, внешнего уровня, из которых впоследствии можно будет вычленить общие части для размещения в отдельных Use-case'ах внутреннего пользования, связываемых с Use-case'ами внешнего уровня отношениями включения или расширения:

Актер: *Оператор*.
Выполняемые действия:

- Задание элементов карты местности.
- Программирование целевых точек.

Актер: *Диспетчер*.
Выполняемые действия:

- Подача команды на начало выполнения маршрутного задания.

- Подача команды на аварийный останов.
- Запрос информации о текущем состоянии подсистем робота.
- Запрос информации о текущем месте расположения робота.

Исходя из вышеперечисленного набора действий, построим начальную Use-case – диаграмму (рис. 3):

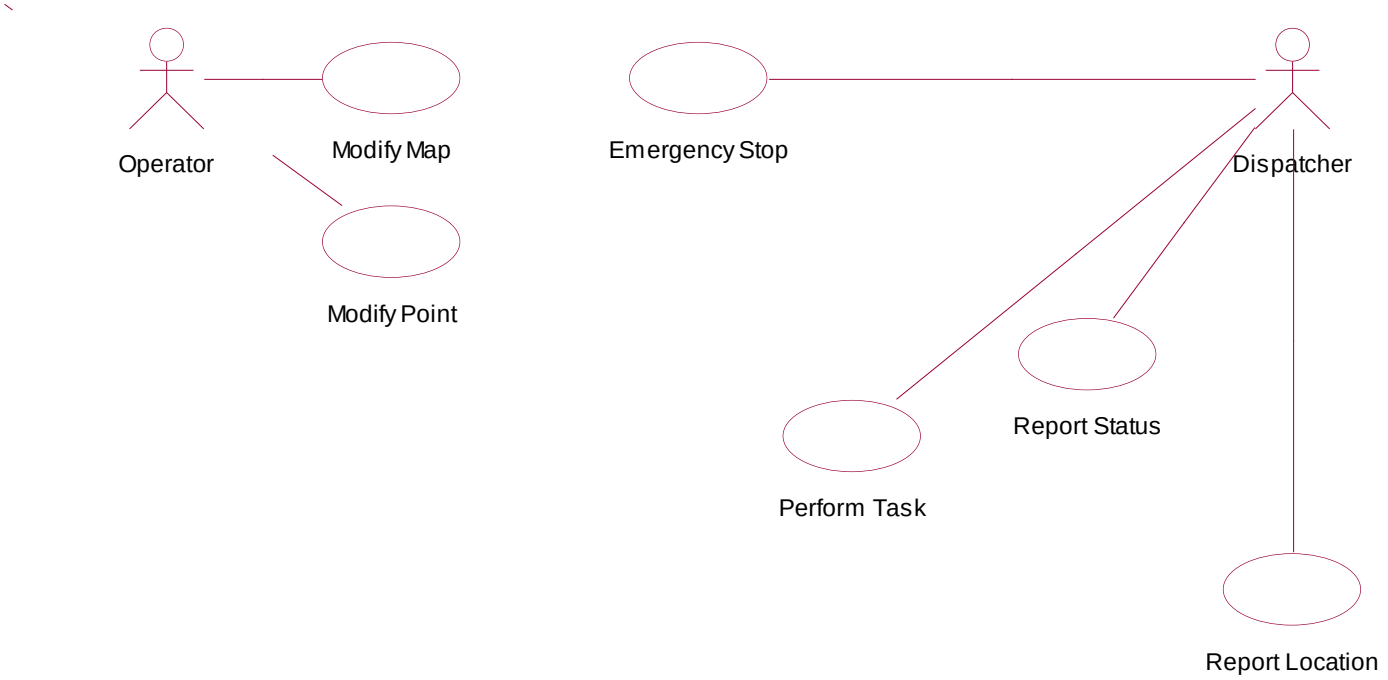

Рис. 3. Начальная Use-case – диаграмма.

Описание Use-case'ов.

Use-case: *ModifyMap.*
Выполняемое действие: *Задание элементов карты местности.*
Данный Use-case предназначен для изменения размеров и содержимого бортовой карты местности. Карта имеет прямоугольную форму заданных размеров и разбита на квадраты, содержимое которых можно редактировать с пульта управления. При этом недопустимо редактирование карты или изменение ее размеров во время движения робота по маршруту, то есть должна обеспечиваться автоматическая остановка робота и сброс маршрутного задания при возникновении запроса на изменение элементов карты местности. Таким образом, в Use-case ModifyMap включается следующий набор действий:

- Выдача команды на аварийный останов
- Непосредственное изменение размеров и/или содержимого карты местности.

Use-case: *ModifyPoint.*
Выполняемое действие: *Программирование целевых точек.*
Данный Use-case предназначен для изменения набора целевых точек. Бортовая система управления располагает фиксированным числом целевых точек, так что процедура программирования целевой точки сводится к выбору одной из точек и заданию координат на карте, соответствующих этой точке. При этом недопустимо программирование целевых точек во время движения робота по маршруту, так как это может привести к нарушению текущего маршрутного задания, то есть должна обеспечиваться автоматическая остановка робота и сброс маршрутного задания при возникновении запроса на программирование целевых точек. Таким образом, в Use-case ModifyMap включается следующий набор действий:

- Выдача команды на аварийный останов
- Непосредственное изменение координат одной из целевых точек.

Use-case: *EmergencyStop.*
Выполняемое действие: *Подача команды на аварийный останов.*
Данный Use-case предназначен для осуществления аварийного останова и сброса текущего маршрутного задания, в ответ на поступивший запрос. Аварийный останов должен выполняться в любой момент времени независимо от состояния и места расположения робота.

Use-case: *ReportLocation.*
Выполняемое действие: *Запрос информации о текущем месте расположения робота.*
Данный Use-case предназначен для осуществления запроса и получения информации о текущем месте расположения робота. Место расположения характеризуется двухэлементным вектором, определяющим текущие координаты робота в контексте и масштабе бортовой карты местности. Выдача информации о текущем месте расположения по запросу должна выполняться в любой момент времени независимо от состояния и места расположения робота.

Use-case: *ReportStatus.*
Выполняемое действие: *Запрос информации о текущем состоянии подсистем робота.*
Данный Use-case предназначен для осуществления запроса и получения информации о текущем состоянии подсистем робота. При поступлении запроса о состоянии подсистем, производится последовательный опрос всех устройств системы для получения значений контролируемых параметров и сравнение полученных данных с эталонными. Выдача информации о текущем состоянии подсистем робота по запросу должна выполняться в любой момент времени независимо от состояния и места расположения робота.

Use-case: *PerformTask.*
Выполняемое действие: *Подача команды на начало выполнения маршрутного задания.*
Данный Use-case предназначен для инициирования процесса выполнения маршрутного задания и его отслеживания вплоть до момента достижения заданной целевой точки на местности. Инициатором начала выполнения данного Use-case'а является диспетчер, взаимодействующий с системой управления роботом посредством дистанционного терминала. В ходе выполнения данного Use-case'а циклически производятся следующие действия вплоть до момента достижения заданной целевой точки на местности:

- Запрос информации о текущем состоянии подсистем робота (проверка возможности дальнейшего движения).
- Запрос информации о текущем месте расположения робота (получение текущих координат).
- Пересчет маршрута (траектории) к заданной целевой точке.
- Выполнение единичного шага для приближения к целевой точке согласно текущему маршруту.
- Запрос информации о текущем состоянии подсистем робота (проверка работоспособности в результате сделанного шага).
- Запрос информации о текущем месте расположения робота (сверка фактических координат с расчетными).

Таким образом, имея описание Use-case'ов и входящих в них действий, можем приступить к процессу разработки конечной Use-case – диаграммы. При этом представляется целесообразным руководствоваться следующими соображениями:

- Use-case'ы ModifyMap, ModifyPoint, PerformTask состоят из набора обособленных действий, в число которых входят такие как *Выдача команды на аварийный останов*, *Запрос информации о текущем состоянии подсистем робота*, *Запрос информации о текущем месте расположения робота*, которые уже определены в качестве отдельных Use-case'ов внешнего уровня, поэтому целесообразно воспользоваться уже существующими Use-case'ами для того, что бы повысить степень повторного использования в системе. При этом данные Use-case'ы связываются с исходными отношениями включения и расширения.
- Use-case Perform task содержит также набор обособленных действий, не определенных ранее. В их число входят *Пересчет маршрута (траектории) к заданной целевой точке* и *Выполнение единичного шага для приближения к целевой точке согласно текущему маршруту*. Данные действия выделяются в отдельные Use-case'ы внутреннего пользования.
- Ввиду того, что действие *Выдача команды на аварийный останов* выполняется только при наступлении определенных условий (невозможность дальнейшего движения и т.д.), определим отношение между данным Use-case'ом и использующим его Use-case'ом PerformTask как отношение расширения.
- Ввиду того, что действия *Запрос информации о текущем состоянии подсистем робота*, *Запрос информации о текущем месте расположения робота, Пересчет маршрута (траектории) к заданной целевой точке* и *Выполнение единичного шага для приближения к целевой точке согласно текущему маршруту* выполняются циклически на каждом шаге работы Use-case'а PerformTask, определим отношение между данным Use-case'ом и Use-case'ами как отношение включения.
- Ввиду того, что действие *Выдача команды на аварийный останов* выполняется, безусловно в рамках Use-case'ов ModifyMap и ModifyPoint, определим отношение между данным Use-case'ом и использующими его Use-case'ами как отношение включения.

В итоге имеем конечную Use-case – диаграмму, отображенную на рис. 4:

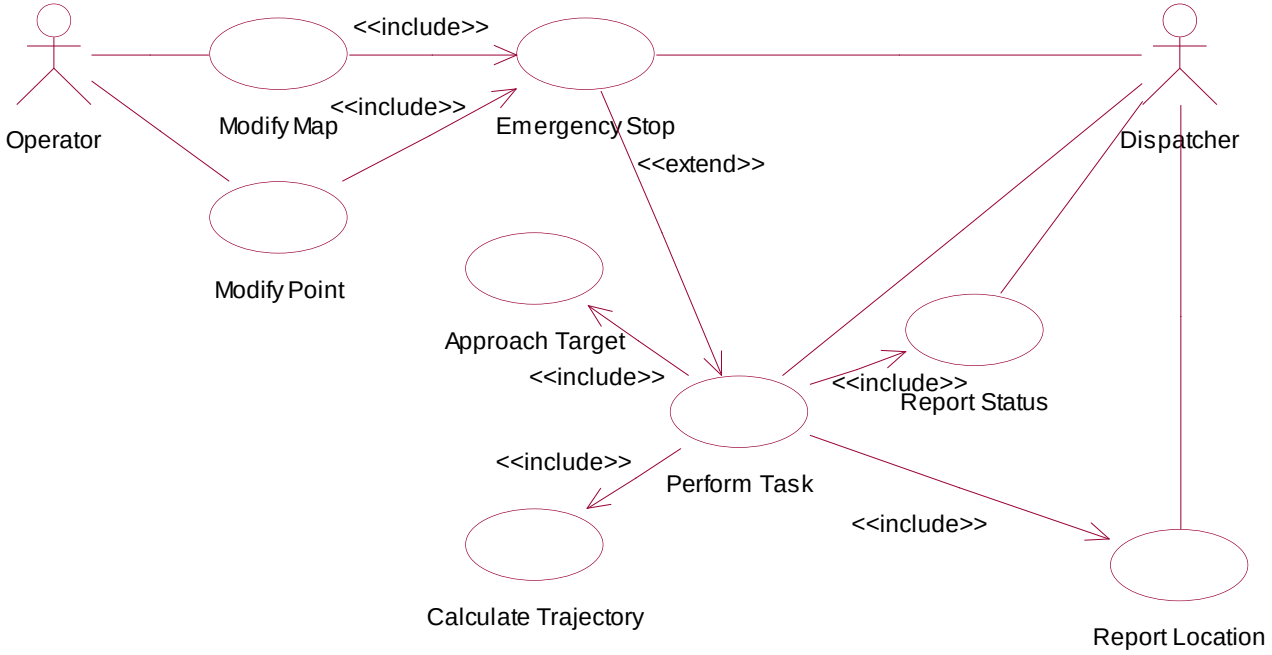

Рис. 4. Конечная Use-case - диаграмма

Схема аппаратной реализации системы управления.

Имея в распоряжении описание Use-case'ов и список входящих в них действий, можем приступить к разработке схемы аппаратной реализации, позволяющей производить заложенные в Use-case – диаграмму действия на уровне совместной работы аппаратных устройств. Схема аппаратной реализации отображена на рис. 5:

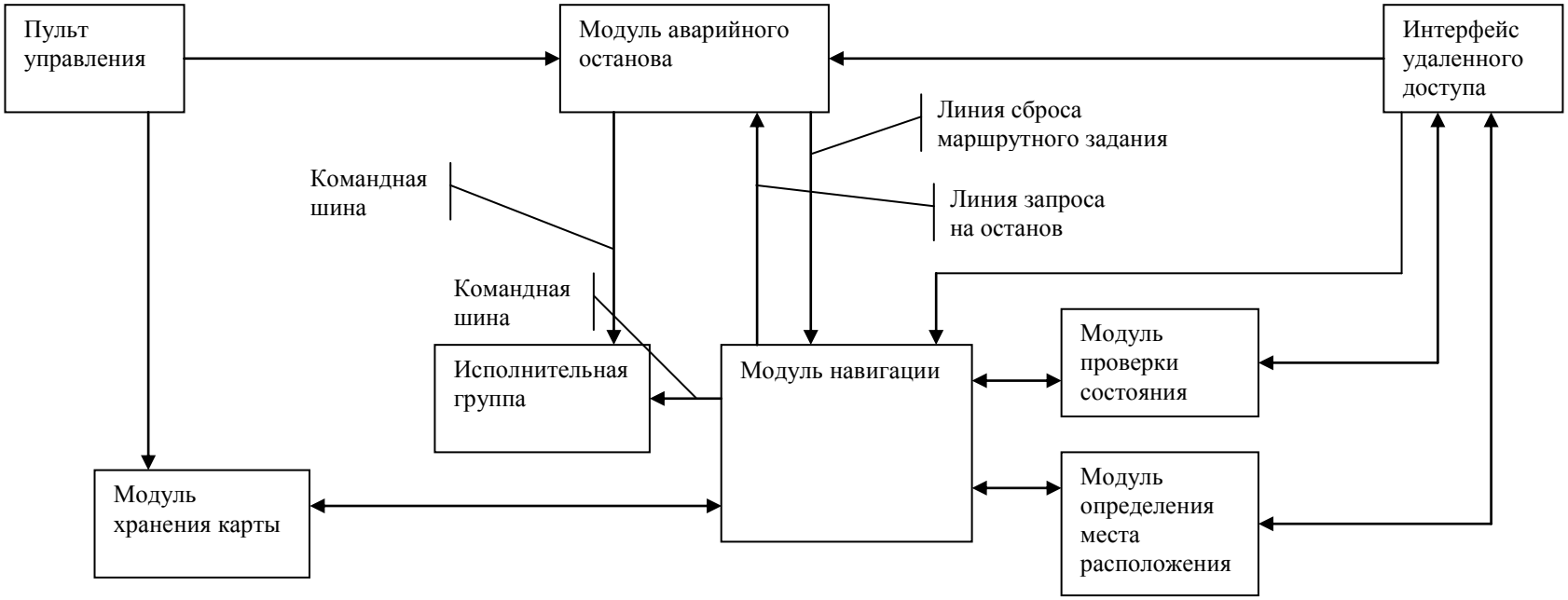

Рис. 5. Схема аппаратной реализации

Описание устройств.

Приведем описание устройств, входящих в схему аппаратной реализации и укажем выполняемые ими функции.

Устройство: *Пульт управления*
Пульт управления установлен на борту мобильного робота и предназначен для произведения оператором действий по изменению размеров и содержимого карты местности и программирования целевых точек. Пульт управления оборудован кнопками управления и многофункциональным дисплеем (МФД) на базе жидкокристаллических индикаторов (ЖКИ). Многофункциональный дисплей предназначен для отображения содержимого карты местности и программируемых целевых точек. Общий вид лицевой панели пульта управления отображен на рис. 6:

Рис. 6. Общий вид лицевой панели пульта управления

Пульт управления соединен с модулем хранения карты с целью обеспечения возможности посылки сигналов для изменения содержимого карты и с блоком аварийного останова с целью обеспечения возможности посылки команды на останов перед началом редактирования карты или программирования целевых точек.

Устройство: *Интерфейс удаленного доступа*
Данное устройство предназначено для обеспечения дистанционного доступа посредством радиоканала от диспетчера. Устройство выполняет функции запроса состояния подсистем, текущего места расположения, а также приема команд на начало выполнения маршрутного задания и осуществление аварийного останова. Интерфейс удаленного доступа соединен с модулем навигации, модулем аварийного останова, модулем проверки состояния и модулем определения места расположения.

Устройство: *Модуль аварийного останова*
Модуль аварийного останова предназначен для посылки исполнительной группе сигнала на немедленный останов робота. Данное устройство работает параллельно с другими, обеспечивая тем самым возможность осуществления аварийного останова в любой момент времени независимо от состояния и места расположения как системы управления, так и робота в целом. Модуль аварийного останова может принимать запросы на аварийный останов от пульта управления в рамках подготовки к процедуре изменения содержимого карты местности или программирования целевых точек, а также от интерфейса удаленного доступа, принимающего запрос на аварийный останов от диспетчера. Соответственно модуль аварийного останова соединен командной шиной с исполнительной группой, принимающей к выполнению команду аварийного останова.

Устройство: *Модуль хранения карты*
Данное устройство представляет собой запоминающее устройство, обеспечивающее хранение карты местности и набора целевых точек для их последующего использования модулем навигации в процессе построения и выполнения маршрутного задания. Модуль хранения карты позволяет хранить фиксированное количество программируемых целевых точек и карту, ограниченную по размерам объемом установленной в модуле памяти. Устройство соединено с модулем навигации, осуществляющем выборку информации о карте местности и целевых точках, а так же с пультом управления, предназначенным для осуществления функций программирования целевых точек и изменения содержимого карты местности.

Устройство: *Модуль проверки состояния*
Модуль проверки состояния предназначен для осуществления функций аппаратного опроса подсистем робота с целью выявления неисправностей или нарушений в работе той или иной подсистемы и принятия решения о невозможности дальнейшего функционирования робота. Данное устройство присоединено к модулю навигации в качестве устройства внутреннего пользования, а также к интерфейсу удаленного доступа для удовлетворения запросов на проверку состояния, поступающих от диспетчера. В данном случае модуль проверки состояния работает как устройство внешнего пользования.

Устройство: *Модуль определения места расположения*
Модуль определения места расположения предназначен для осуществления функций сопоставления истинного места расположения с точкой на карте с целью вычисления нового маршрутного задания на каждом цикле работы модуля навигации либо проверки корректности выполнения единичного шага маршрутного задания. Данное устройство присоединено к модулю навигации в качестве устройства внутреннего пользования, а также к интерфейсу удаленного доступа для удовлетворения запросов на определения места расположения, поступающих от диспетчера. В данном случае модуль определения места расположения работает как устройство внешнего пользования.

Устройство: *Модуль навигации*
Модуль навигации представляет собой центрально вычислительное устройство, предназначенное для осуществления функций разработки маршрутного задания и посылки управляющих команд с целью выполнения единичного шага разработанного маршрутного задания. Данное устройство также предназначено для циклического опроса модуля проверки состояния и модуля определения места расположения с целью принятия решения о возможности или невозможности дальнейшего движения робота, а также определения корректности выполнения единичного шага маршрутного задания. Модуль навигации соединен с модулем проверки состояния, модулем определения места расположения, интерфейсом удаленного доступа, модулем хранения карты местности, исполнительной группой посредством командной шины, а также модулем аварийного останова двумя противоположно направленными сигнальными линиями для посылки запроса на аварийный останов от модуля навигации к модулю аварийного останова и посылки сигнала сброса текущего маршрутного задания в результате осуществления аварийного останова.

Устройство: *Исполнительная группа*
Исполнительная группа представляет собой набор аппаратных контроллеров, подключенных к общей командной шине и осуществляющих преобразование логических команд, получаемых от модуля навигации и модуля аварийного останова в команды, которые позволяют осуществлять непосредственное управление движущей частью робота. Исполнительная группа

присоединена к модулю навигации и модулю аварийного останова посредством командных шин, предназначенных для посылки команд на выполнение.

Разработанная схема аппаратной реализации, наряду с имеющимися Use-case'ами, позволяет значительно облегчить дальнейшую разработку диаграмм последовательности на этапе Развитие, существенно снижая степень неопределенности в процессе разработки, что способствует созданию диаграмм последовательности с минимальным числом ошибок и неверных конструктивных решений.

Этап Развитие.

На данном этапе разрабатываются сценарии работы программной системы для каждого из Use-case'ов, строятся диаграммы последовательности для каждого из сценариев, на основании объектов, входящих в диаграммы последовательности, определяются их абстракции – классы и осуществляется планирование итераций этапа Конструирование.

Определение сценариев.

Определим независимые (т.е. инициируемые раздельно) потоки, входящие в состав каждого из разработанных на предыдущем этапе Use-case'ов.

Use-case: *ModifyMap.*
Выполняемое действие: *Задание элементов карты местности.*
В данный Use-case входят два независимых потока, предназначенных для работы с картой местности:
- Изменение размеров карты.
- Изменение содержимого карты.

Use-case: *ModifyPoint.*
Выполняемое действие: *Программирование целевых точек.*

Данный Use-case содержит только один поток, связанный с выбором и изменением одной из программируемых целевых точек, сопровождаемых предварительной выдачей команды аварийного останова с целью избежать нарушения корректности работы модуля навигации в ходе выполнения маршрутного задания, и, как следствие такого нарушения, потери управляемости роботом.

Use-case: *EmergencyStop.*
Выполняемое действие: *Подача команды на аварийный останов.*
Данный Use-case содержит только один поток, связанный с непосредственной подачей команды модулем аварийного останова исполнительной группе по специально предназначенной для этих целей командной шине.

Use-case: *ReportLocation.*
Выполняемое действие: *Запрос информации о текущем месте расположения робота.*
Данный Use-case содержит только один поток, связанный с опросом модуля определения места расположения.

Use-case: *ReportStatus.*
Выполняемое действие: *Запрос информации о текущем состоянии подсистем робота.*
Данный Use-case содержит только один поток, связанный с опросом модуля проверки состояния.

Use-case: *PerformTask.*
Выполняемое действие: *Подача команды на начало выполнения маршрутного задания.*
Данный Use-case содержит два потока, предназначенных для выполнения функции передвижения робота по направлению к выбранной цели:

- Выбор целевой точки
- Движение к выбранной целевой точке.

Каждое из вышеперечисленных действий представляет собой отдельный сценарий, реализуемый в рамках одного Use-case’а.

Разработка диаграмм последовательности.

Разработку диаграмм последовательности произведем на основании вышеприведенной схемы функционирования аппаратного обеспечения системы управления роботом.
Разработка ведется отдельно для каждого Use-case'а и сценария:

Use-case: *ModifyMap**, сценарий** Изменение размеров карты*
В рамках данного сценария оператор производит изменение размеров карты, находящейся в модуле хранения карты посредством пульта управления.
Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 7):

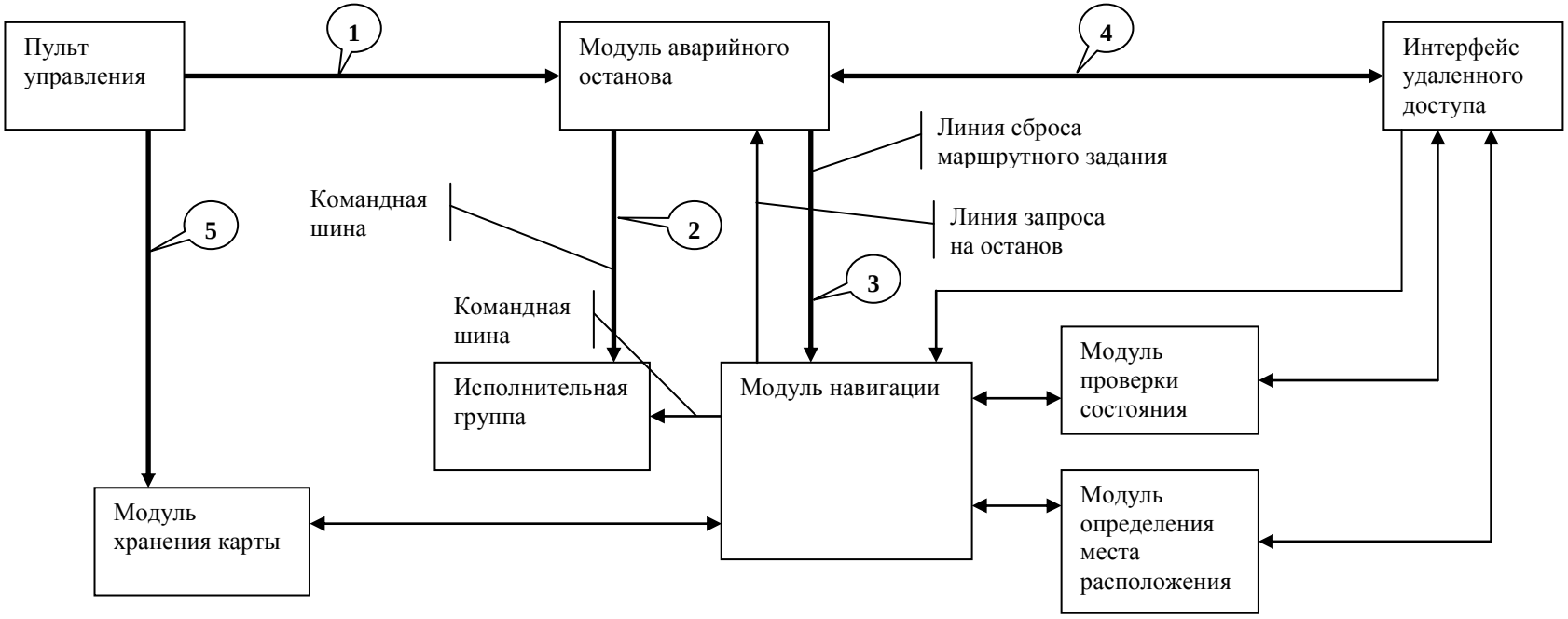

Рис. 7. Сценарий Изменение размеров карты

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Пульт управления посылает модулю аварийного останова запрос на аварийный останов.
2. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.
3. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
4. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.
5. Пульт управления посылает модулю хранения карты запрос на изменение размеров карты местности.

Use-case: *ModifyMap**, сценарий** Изменение содержимого карты*

В рамках данного сценария оператор производит изменение содержимого карты, находящейся в модуле хранения карты посредством пульта управления.

Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 8):

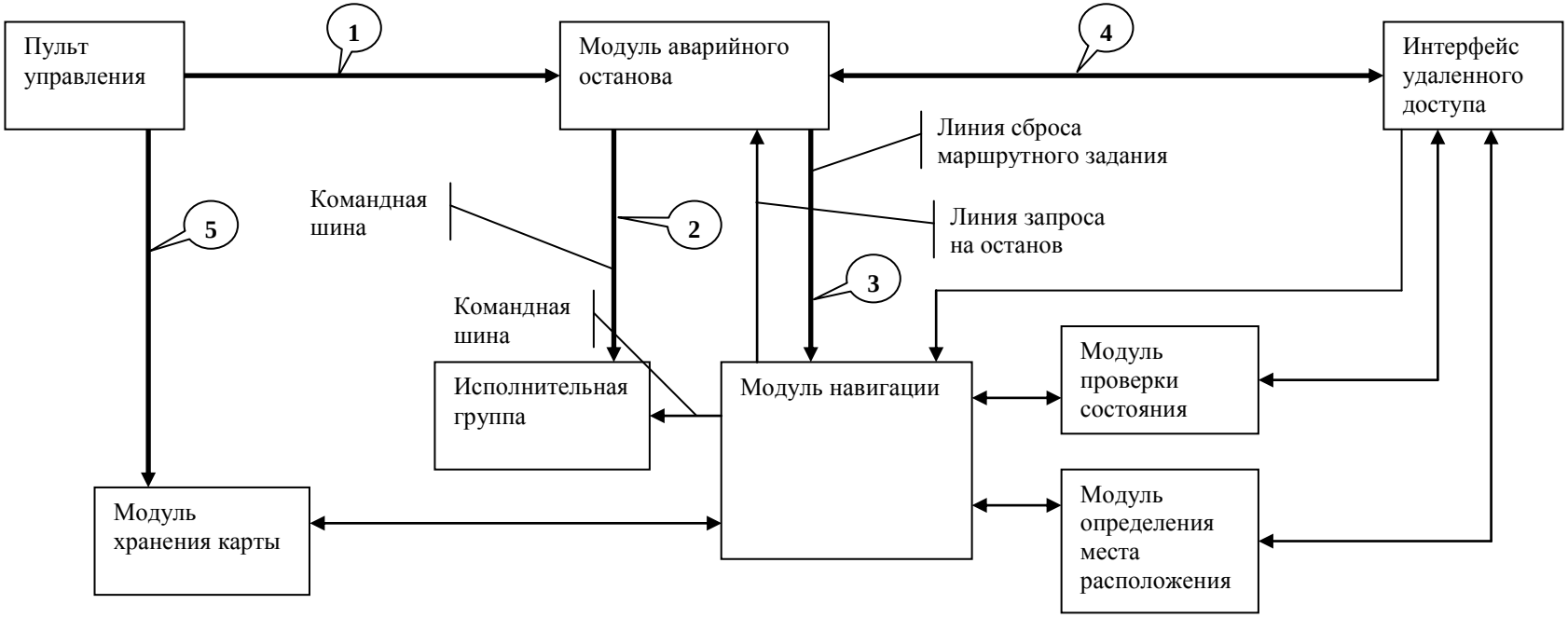

Рис. 8. Сценарий Изменение содержимого карты

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Пульт управления посылает модулю аварийного останова запрос на аварийный останов.
2. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.
3. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
4. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.
5. Пульт управления посылает модулю хранения карты запрос на изменение содержимого карты местности.

Построим диаграмму последовательности для двух вышеприведенных сценариев (рис. 9):

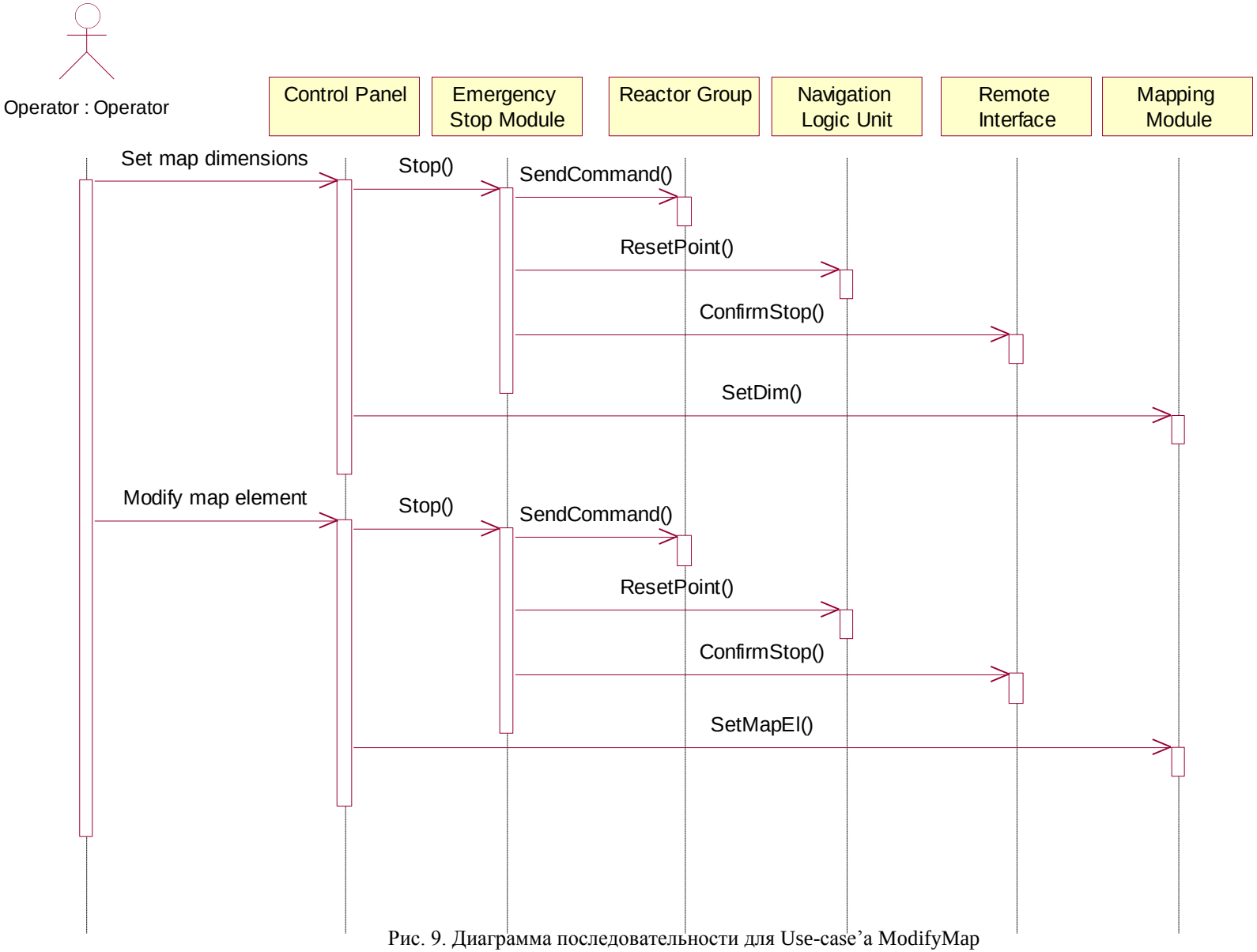

Рис. 9. Диаграмма последовательности для Use-case'а ModifyMap

Use-case: *ModifyPoint*

В рамках данного сценария оператор производит программирование целевых точек, находящихся в модуле хранения карты посредством пульта управления.

Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 10):

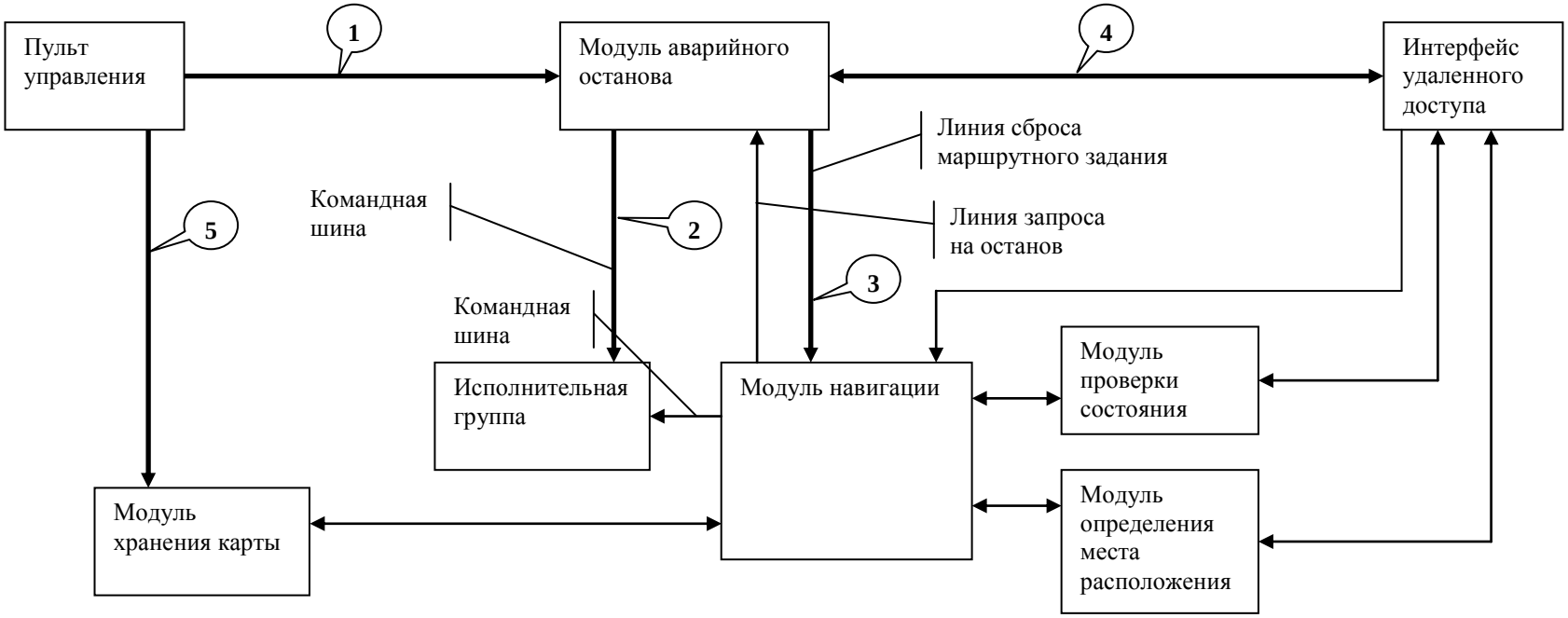

Рис. 10. Use-case ModifyPoint

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Пульт управления посылает модулю аварийного останова запрос на аварийный останов.
2. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.
3. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
4. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.

5. Пульт управления посылает модулю хранения карты запрос на изменение содержимого карты местности.

Построим диаграмму последовательности для вышеприведенного сценария (рис. 11):

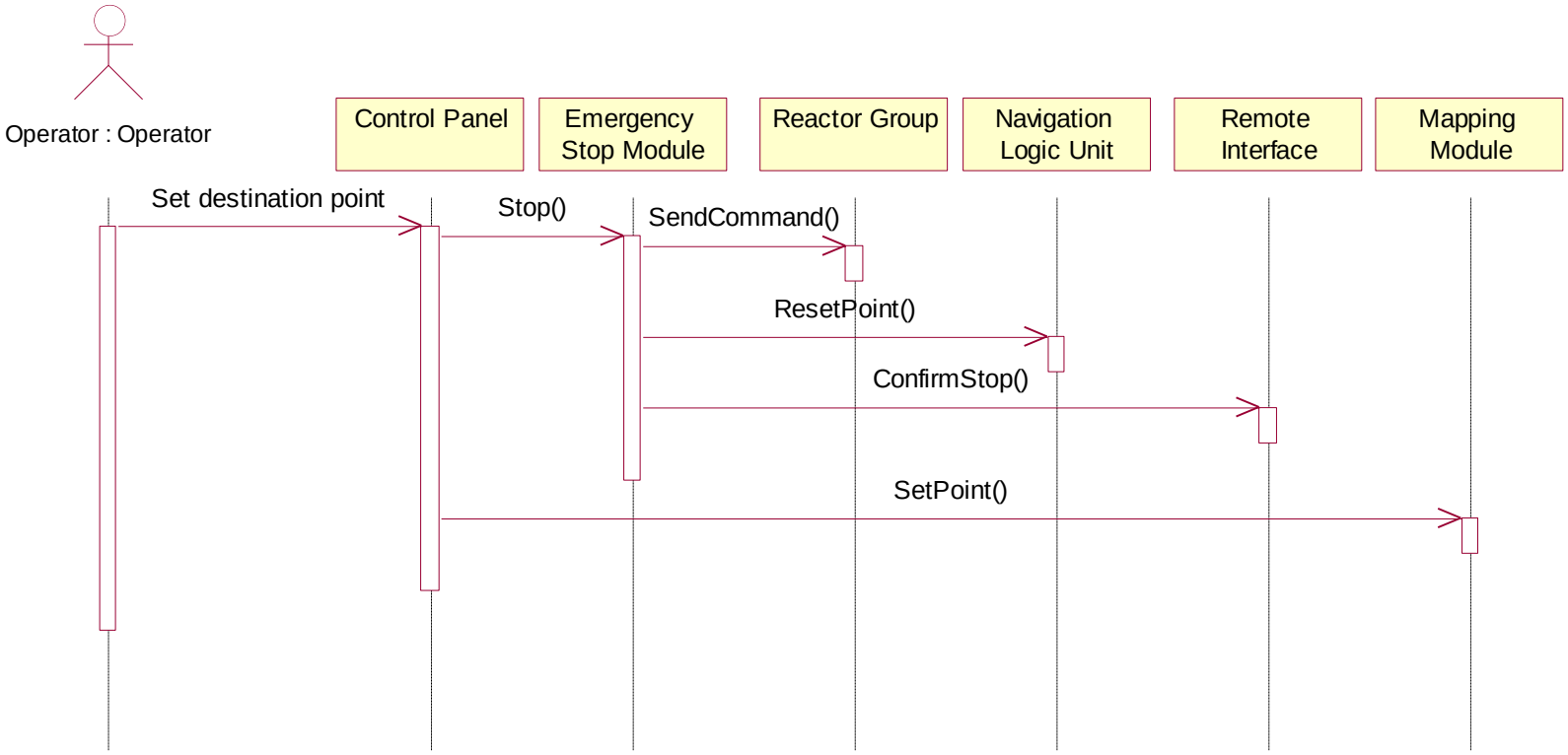

Рис. 11. Диаграмма последовательности для Use-case'а ModifyPoint

Use-case: *EmergencyStop*

В рамках данного сценария производится посылка модулем аварийного останова команды исполнительной группе. Также происходит сброс текущей целевой точки в модуле навигации и извещение диспетчера об аварийном останове через интерфейс удаленного доступа.

Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 12):

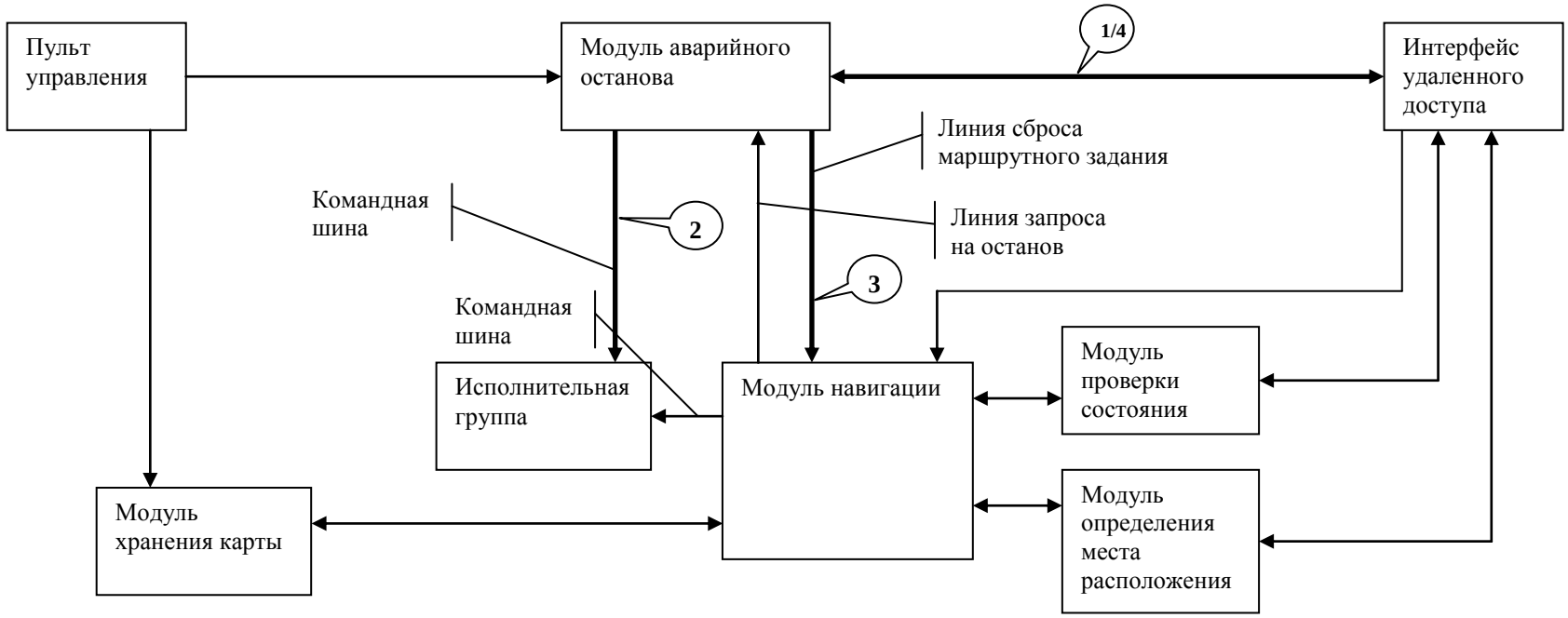

Рис. 12. Use-case EmergencyStop

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Интерфейс удаленного доступа извещает модуль аварийного останова о необходимости совершить аварийный останов.
2. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.
3. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
4. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.

Построим диаграмму последовательности для вышеприведенного сценария (рис. 13):

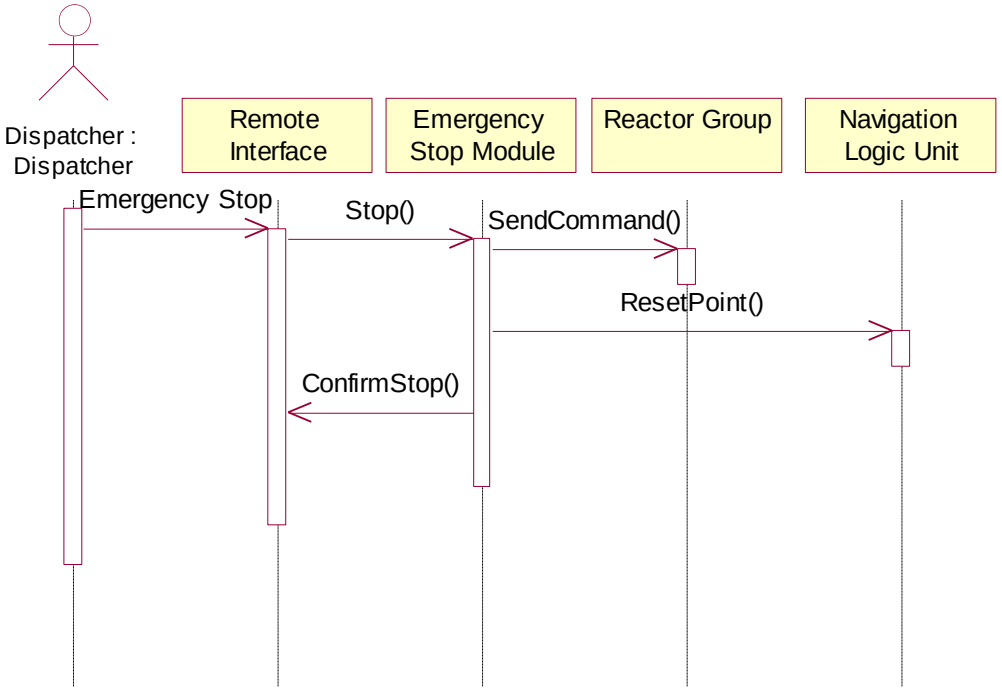

Рис. 13. Диаграмма последовательности для Use-case'а EmergencyStop

Use-case: *ReportLocation*
В рамках данного сценария производится запрос о текущем месте расположения робота.
Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 14):

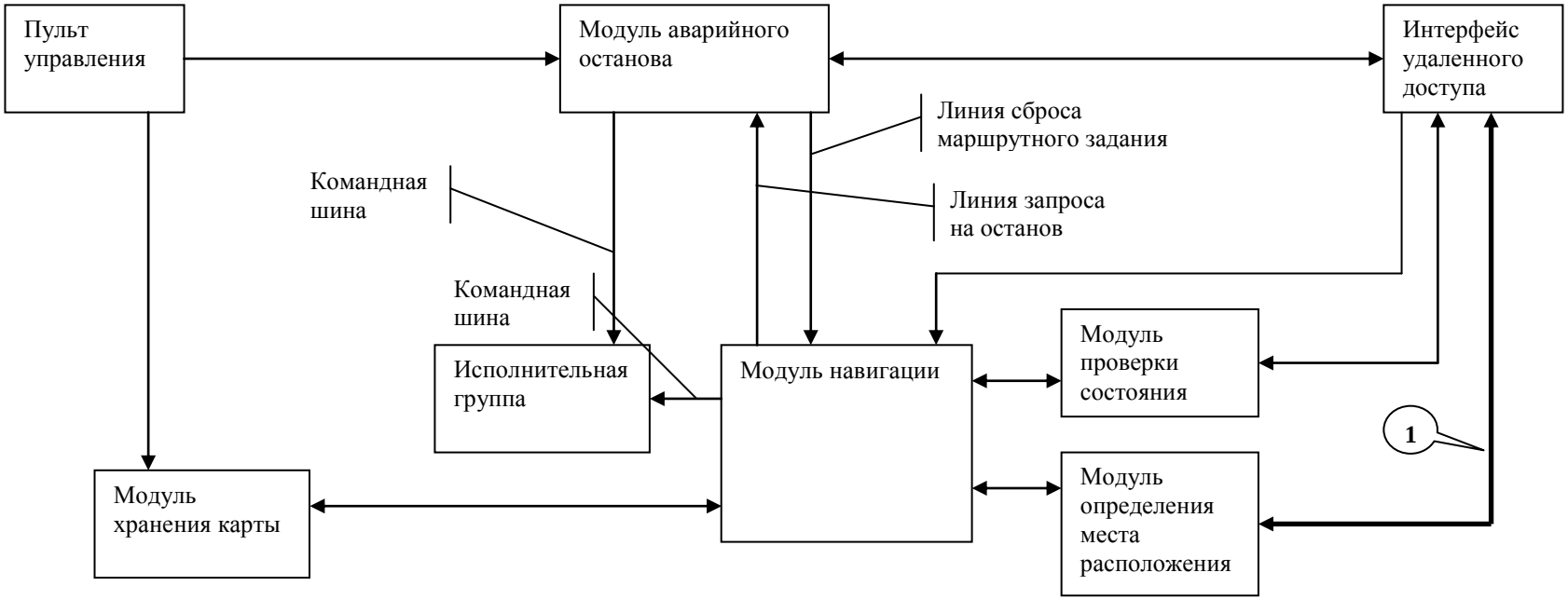

Рис. 14. Use-case ReportLocation

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:
1. Интерфейс удаленного доступа запрашивает информацию о текущем месте расположения у модуля определения места расположения.

Построим диаграмму последовательности для вышеприведенного сценария (рис. 15):

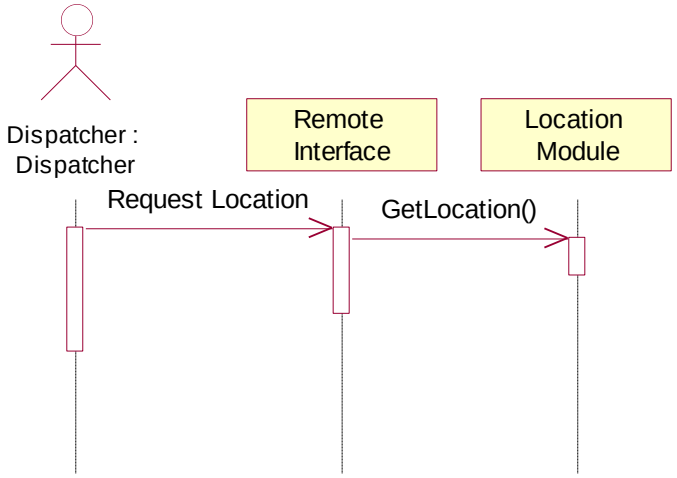

Рис. 15. Диаграмма последовательности для Use-case'а ReportLocation

Use-case: *ReportStatus*

В рамках данного сценария производится запрос о текущем состоянии подсистем робота.

Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 16):

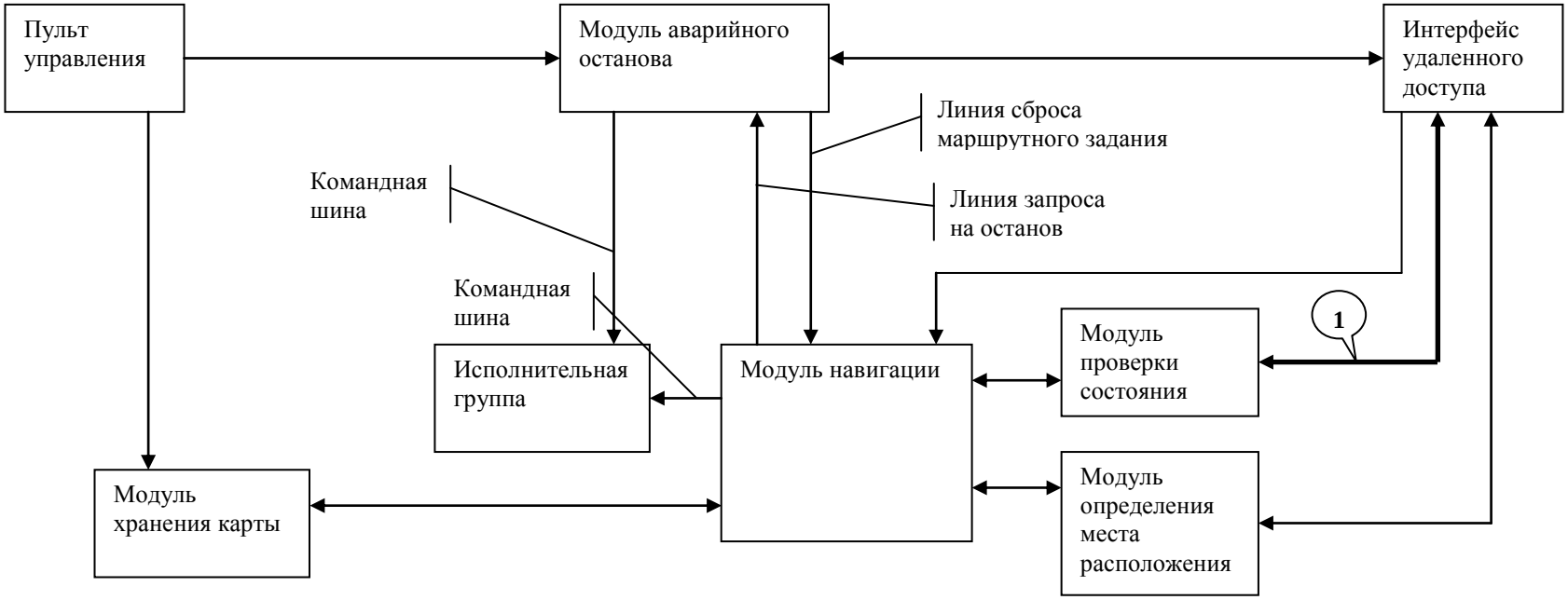

Рис. 16. Use-case ReportStatus

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Интерфейс удаленного доступа запрашивает информацию о текущем состоянии подсистем робота у модуля проверки состояния.

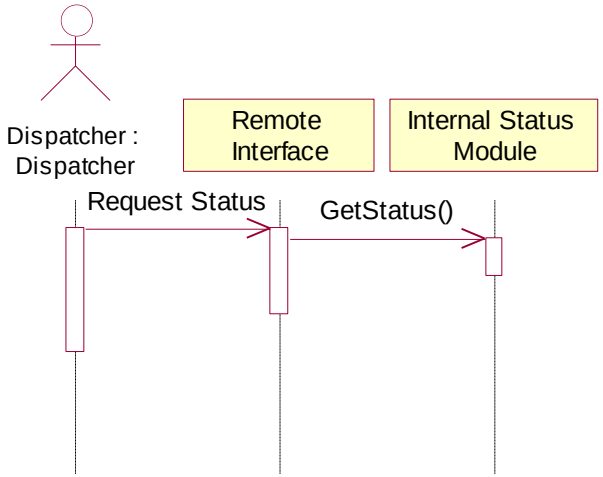

Рис. 17. Диаграмма последовательности для Use-case'а ReportLocation

Построим диаграмму последовательности для вышеприведенного сценария (рис. 17):

Use-case: *PerformTask**, сценарий** Выбор целевой точки*
В рамках данного сценария производится выбор целевой точки для ее дальнейшего использования в сценарии Движение к выбранной целевой точке. При выборе точки по ее порядковому номеру производится запрос координат выбираемой точки.

Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 18):

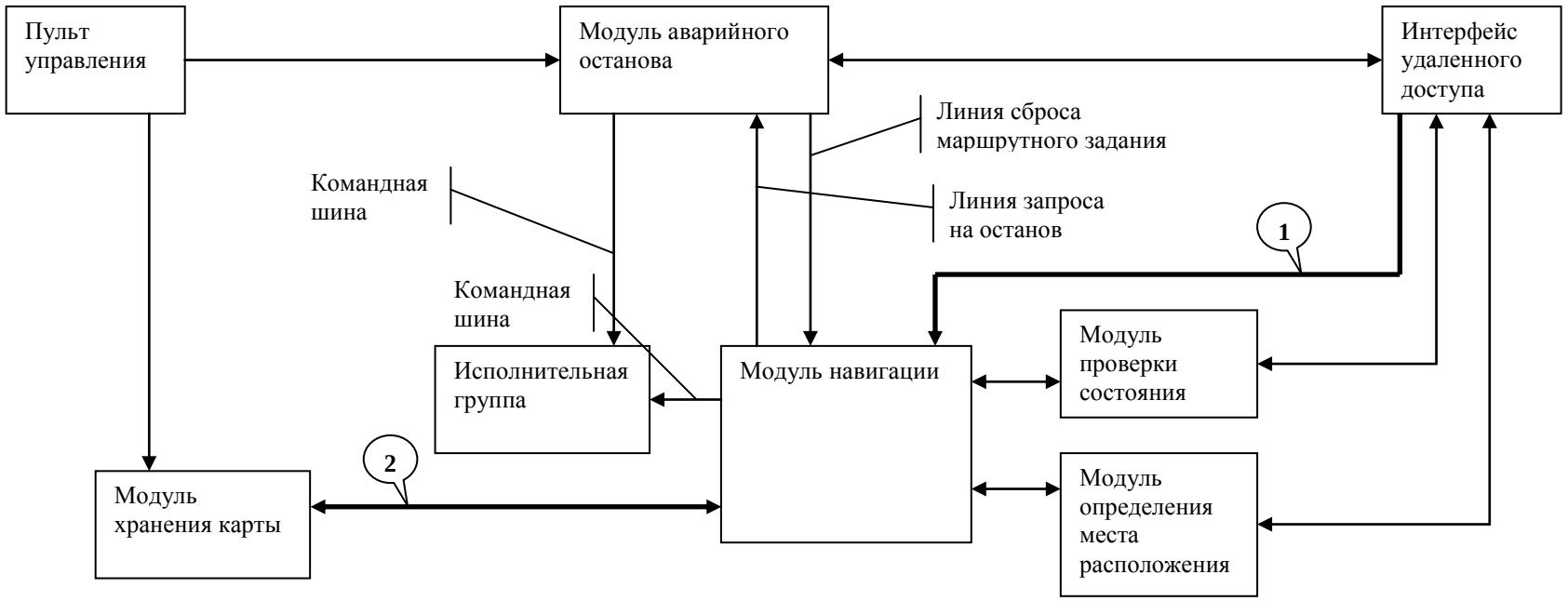

Рис. 18. Сценарий Выбор целевой точки.

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Интерфейс удаленного доступа извещает модуль навигации о том, что диспетчер выбирает новую целевую точку по ее порядковому номеру следования в памяти модуля хранения карты.
2. Модуль навигации производит запрос координат новой точки в модуле хранения карты. Данные координаты запоминаются в модуле навигации для дальнейшего использования.

Use-case: *PerformTask**, сценарий** Движение к выбранной целевой точке*

В рамках данного сценария производится пошаговое движение робота к заданной целевой точке. При этом на каждом шаге происходит пересчет маршрутного задания для обеспечения способности робота гибко реагировать на изменяющуюся обстановку.

Ниже приведена схема задействования аппаратных ресурсов при работе данного сценария (рис. 19):

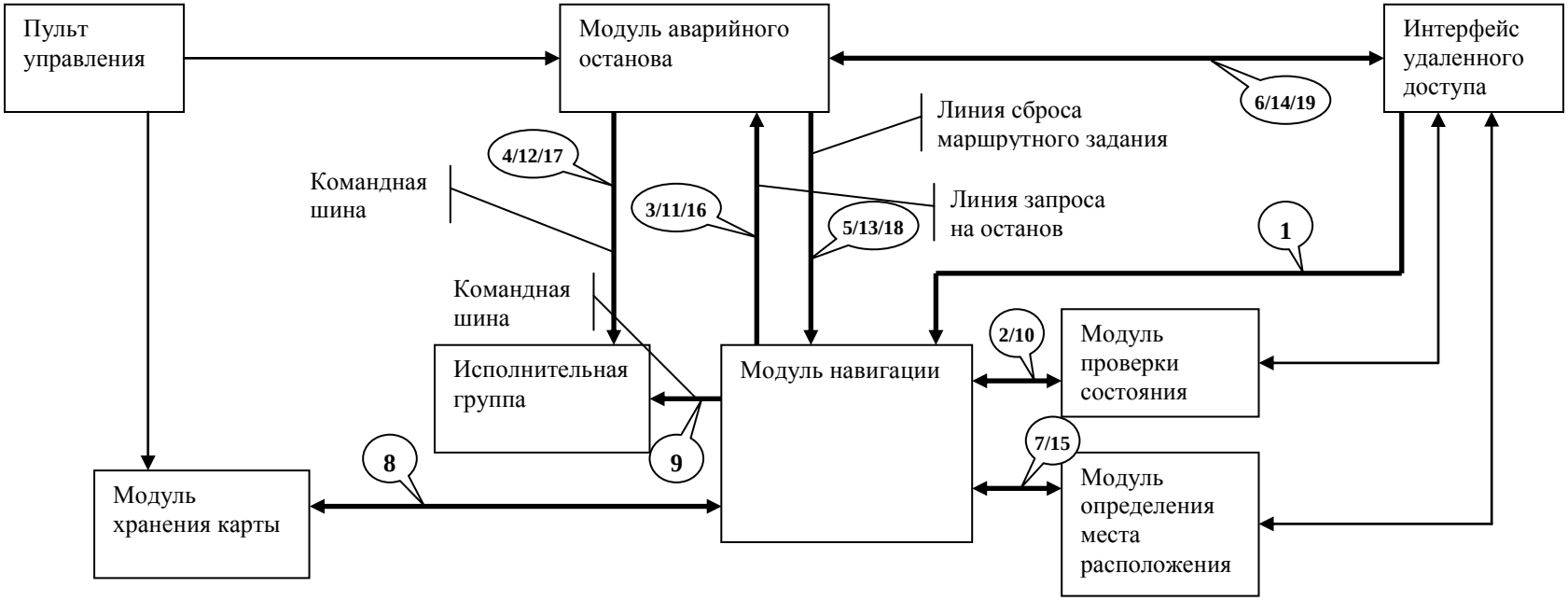

Рис. 19. Сценарий Движение к выбранной целевой точке.

На приведенной диаграмме указаны следующие шаги в порядке их вхождения в рассматриваемый сценарий:

1. Интерфейс удаленного доступа извещает модуль навигации о том, что диспетчер запрашивает начало движения к выбранной целевой точке.
2. Модуль навигации осуществляет проверку целостности робота при помощи модуля проверки состояния для принятия решения о возможности дальнейшего движения. В случае нарушений в работе выполняются шаги 3, 4, 5, 6, иначе – 7 и далее.
3. Модуль навигации извещает модуль аварийного останова о необходимости совершить аварийный останов.
4. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.

5. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
6. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.
7. Модуль навигации запрашивает у модуля определения места расположения текущие координаты робота.
8. Модуль навигации запрашивает у модуля хранения карты необходимые для пересчета маршрутного задания элементы карты местности.
9. После пересчета маршрутного задания, модуль навигации посылает управляющие команды исполнительной группе.
10. Модуль навигации осуществляет повторную проверку целостности робота при помощи модуля проверки состояния для принятия решения о возможности дальнейшего движения в результате произведенных исполнительной группой действий. В случае нарушений в работе выполняются шаги 11, 12, 13, 14, иначе – 15 и далее.
11. Модуль навигации извещает модуль аварийного останова о необходимости совершить аварийный останов.
12. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.
13. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
14. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.
15. Модуль навигации повторно запрашивает у модуля определения места расположения текущие координаты робота для сверки их с расчетными. В случае отклонения от траектории выполняются шаги 16, 17, 18, 19.
16. Модуль навигации извещает модуль аварийного останова о необходимости совершить аварийный останов.
17. Модуль аварийного останова посылает логическую команду аварийного останова исполнительной группе.
18. Модуль аварийного останова сбрасывает текущую целевую точку и текущее маршрутное задание в модуле навигации.
19. Модуль аварийного останова извещает диспетчера через интерфейс удаленного доступа о совершении аварийного останова.

Построим диаграмму последовательности для вышеприведенного сценария (рис. 20):

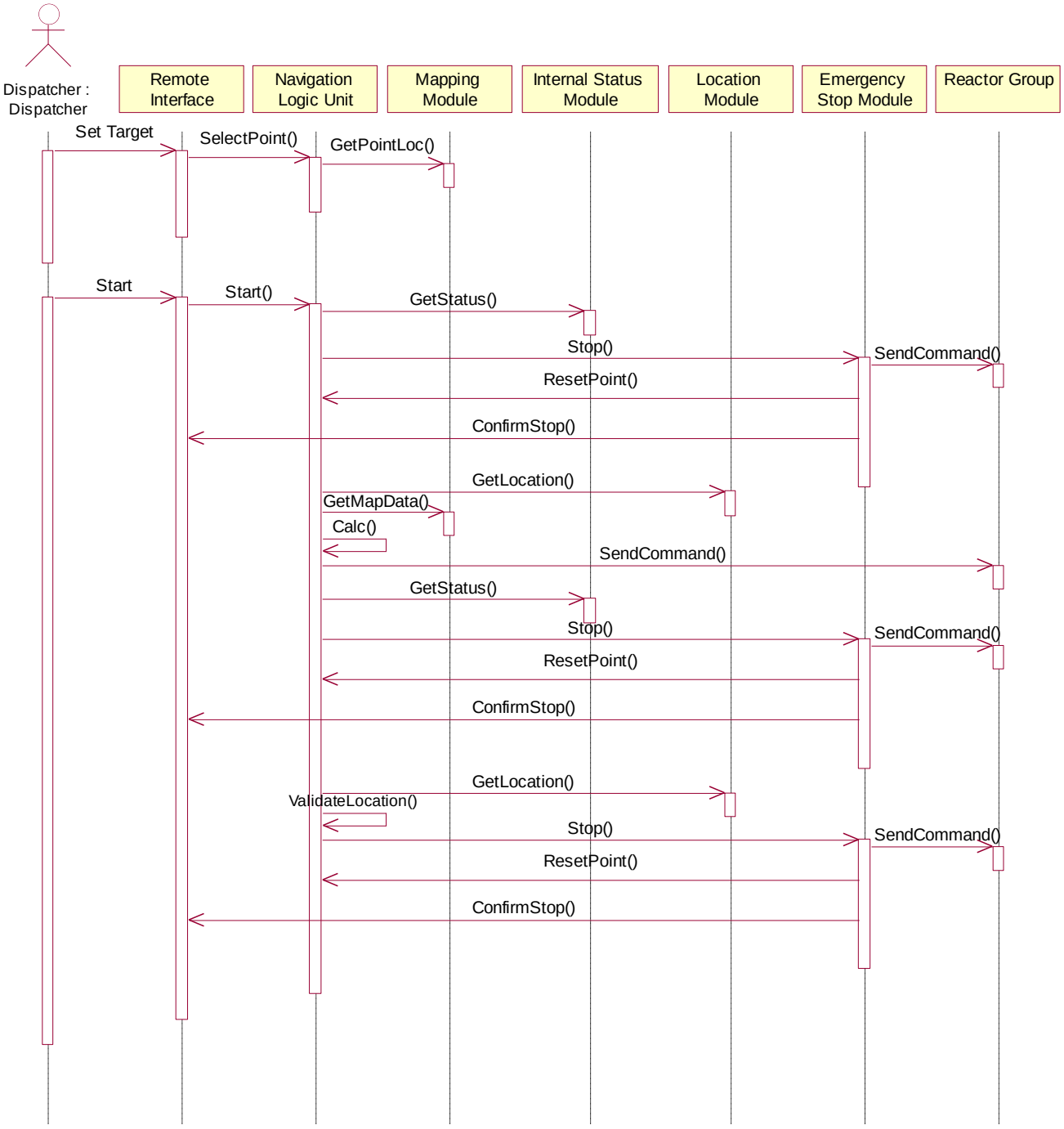

Рис. 20. Диаграмма последовательности для Use-case'а PerformTask

Создание классов.

На основании разработанных диаграмм последовательностей возможно произвести создание классов. При этом классы рассматриваются как абстракции объектов (в контексте данной задачи – аппаратных устройств), задействованных в диаграммах последовательности.

Класс **CControlPanel**
Данный класс описывает устройство *Пульт управления* (на диаграммах последовательности – Control Panel) и предназначен для осуществления взаимодействия с оператором, выполняющим функции изменения карты местности и программирования целевых точек.

Класс **CRemoteInterface**
Данный класс описывает устройство *Интерфейс удаленного доступа* (на диаграммах последовательности – Remote Interface) и предназначен для обеспечения взаимодействия с диспетчером, производящим инициирование выполнения маршрутного задания и прочие действия.

Класс **CIntStatusModule**
Данный класс описывает устройство *Модуль проверки состояния* (на диаграммах последовательности – Internal Status Module) и предназначен для сбора и выдачи информации о состоянии аппаратных подсистем робота с целью обеспечения возможности принятия решения о возможности или невозможности дальнейшего движения.

Класс **CLocationModule**
Данный класс описывает устройство *Модуль определения места расположения* (на диаграммах последовательности – Location Module) и предназначен для выдачи по запросу интерфейса удаленного доступа или модуля навигации текущих координат робота.

Класс **CMappingModule**
Данный класс описывает устройство *Модуль хранения карты* (на диаграммах последовательности – Mapping Module) и предназначен для хранения и обеспечения доступа к карте местности. Данный класс также предназначен для хранения координат программируемых целевых точек.

Класс **CEmStopModule**
Данный класс описывает устройство *Модуль аварийного останова* (на диаграммах последовательности – Emergency Stop Module) и предназначен для посылки команды аварийного останова исполнительной группе по запросу, исходящему от модуля навигации, пульта управления или интерфейса удаленного доступа.

Класс **CNavLogicUnit**
Данный класс описывает устройство *Модуль навигации* (на диаграммах последовательности – Navigation Logic Unit) и предназначен для осуществления управлением роботом через исполнительную группу согласно вычисляемому на каждом такте работы модуля маршрутному заданию.

Класс **CReactorGroup**
Данный класс описывает устройство *Исполнительная группа* (на диаграммах последовательности – Reactor Group) и предназначен для обеспечения взаимодействия между устройствами-источниками логических команд (модуль навигации и модуль аварийного останова) с аппаратными портами контроллеров приводов робота. В обязанности данного класса также входит дешифрация поступающей логической команды и ее декомпозиция на набор аппаратно-зависимых инструкций, воспринимаемых контроллерами исполнительной группы.

Как видно из диаграмм последовательности, некоторые вышеприведенные устройства обладают набором общих функциональных особенностей. В частности, устройства Модуль навигации и Модуль аварийного останова посылают команды исполнительной группе и соединены с ней посредством специальной командной шины. В терминах разрабатываемых классов это означает, что оба класса, соответствующие вышеприведенным устройствам должны располагать ссылкой на объект класса CReactorGroup и вызывать его метод SendCommand(). Представляется целесообразным выделить данные функциональные особенности в отдельный абстрактный класс, наследниками которого являются классы CNavLogicUnit и CEmStopModule:

Класс **CControllerDevice**
Абстрактный класс, являющийся суперклассом по отношению к классам CNavLogicUnit и CEmStopModule и реализующим функцию связи устройства с исполнительной группой посредством командной шины.

Также, в целях повышения расширяемости программной системы и облегчения ее отладки, целесообразно ввести дополнительный абстрактный класс, являющийся общим предком для всех классов-абстракций аппаратных устройств,

добавив в него функцию включения/выключения устройства. Такой подход позволит моделировать на процессе разработки ситуации, когда система функционирует лишь частично.

Класс **CDevice**
Абстрактный класс, являющийся суперклассом по отношению ко всем классам, описывающим аппаратные устройства, входящие в состав системы управления. Данный класс осуществляет включение и выключение описываемого устройства.

Также в рамках реализуемой программной модели системы управления необходим отдельный класс, выполняющий роль контейнера, содержащего все остальные устройства системы управления. С точки зрения программной реализации, данный класс содержит в качестве своих полей данных по одному экземпляру объектов каждого из классов, являющихся абстракциями устройств (CControlPanel, CRemoteInterface, CIntStatusModule, CLocationModule, CMappingModule, CEmStopModule, CNavLogicUnit, CReactorGroup).

Класс: **CMobileRobot**
Данный класс описывает шасси, несущее устройства системы управления, и предназначен для хранения экземпляров классов-устройств системы управления.

Учитывая то, что система управления работает в двухмерном пространстве и описывает все точки в виде двухмерного вектора, целесообразно ввести дополнительный класс-запись, хранящий в своих полях-атрибутах два числа, характеризующих точку на плоскости. Экземпляры данного класса могут быть использованы в виде полей-атрибутов или локальных переменных такими классами как CNavLogicUnit, CMappingModule и CLocationModule.

Класс: **CPoint**
Данный класс описывает точку в двухмерном пространстве и предназначен для использования классами CNavLogicUnit, CMappingModule и CLocationModule.

На основании произведенного построения классов можно вывести предварительную диаграмму классов, описывающую методы классов и отношения наследования между отдельными классами (рис. 21):

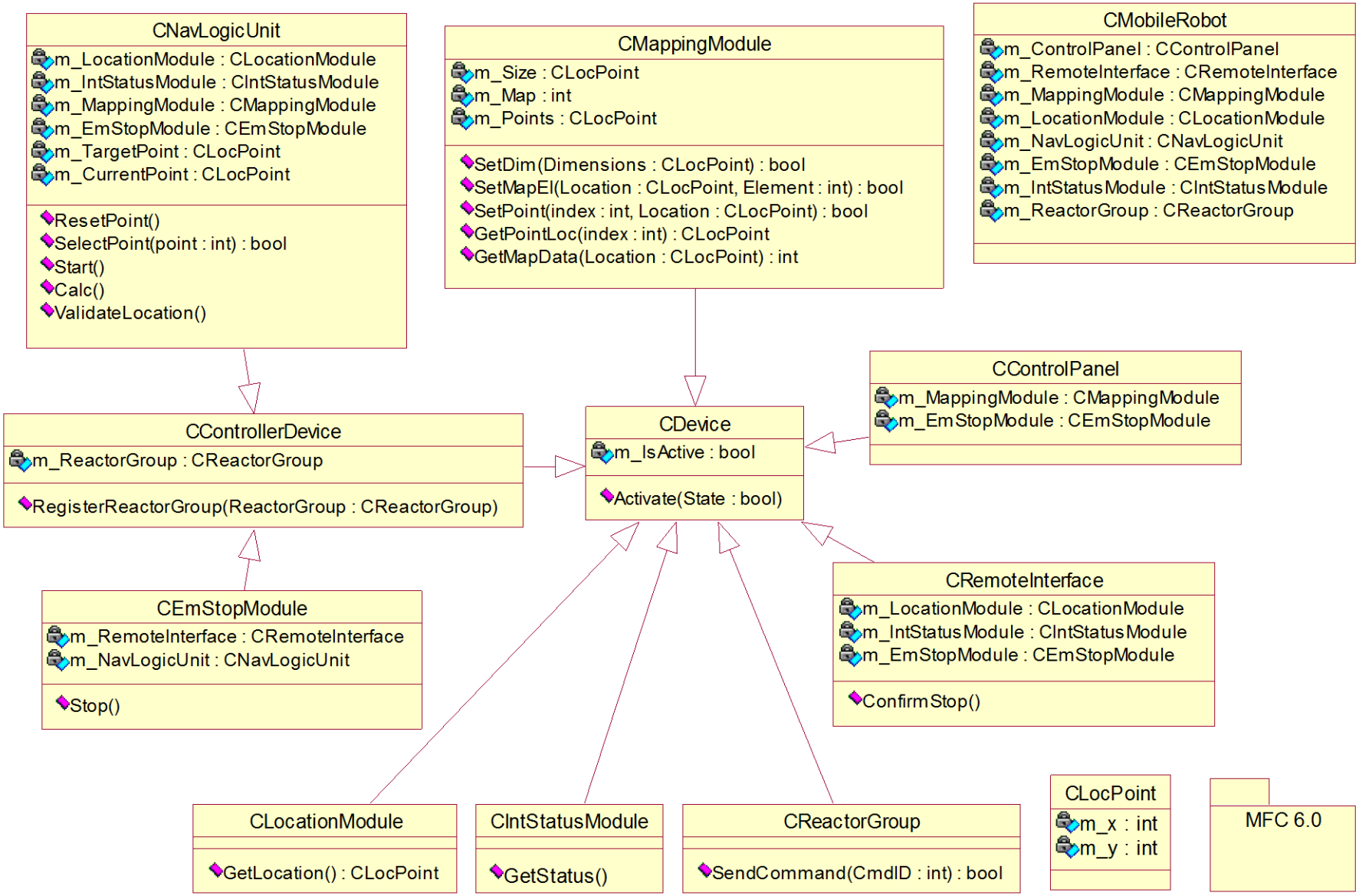

Рис. 21. Предварительная диаграмма классов.

Планирование итераций конструирования

На данной стадии выполнения работ производится разбиение процесса конструирования на отдельные итерации с целью обеспечения возможности управления риском в процессе разработки, а также для определения очередности выполнения работ с точки зрения архитектурного построения программной системы. Обычно планирование итераций производится путем разбиения общего объема работ по Use-case'ам и, если требуется, далее по сценариям, входящим в отдельные Use-case'ы. Ввиду того, что в данной работе производится создание аппаратно-зависимого программного обеспечения, в ходе планирования итераций необходимо также руководствоваться соображениями согласованности разрабатываемой программной системы с архитектурой аппаратного построения системы управления.

Отметим, что между определенными Use-case'ами существует причинно-следственная зависимость. Обозначим эти случаи более подробно. Use-case PerformTask не может полноценно выполняться без реализованных Use-case'ов ReportStatus, ReportLocation, EmergencyStop так как он связан с ними отношениями включения и расширения и использует их в своей работе в качестве Use-case'ов внутреннего пользования. В свою очередь, Use-case'ы ModifyMap и ModifyPoint используют в своей работе Use-case EmergencyStop. Кроме этих очевидных по Use-case – диаграмме причинно-следственных зависимостей, существуют и другие, проистекающие из характера работы системы управления в целом. Очевидно, что система управления не имеет возможности адекватно выполнять какое-либо маршрутное задание без наличия хранимой в памяти карты местности и координат целевых точек. Отсюда следует, что Use-case'ы ModifyMap и ModifyPoint должны предшествовать в процессе конструирования Use-case'у PerformTask, в противном случае после выполнения итерации, реализующей Use-case PerformTask будет получена неработоспособная система, что приведет к невозможности проведения адекватных отладочных работ и значительно увеличит риск разработки программной системы как на данной итерации, так и на всем этапе конструирования. При разработке сложных программных систем чрезвычайно важно проводить отладочные работы параллельно разработке классов, что бы не накапливать ошибки и неверные проектные решения до стадии, когда их исправление будет означать необходимость полной переработки всей программной системы. Исходя из вышеуказанных соображений, целесообразно расположить реализуемые Use-case'ы в следующем порядке:

Итерация 1:

- EmergencyStop
- ReportStatus
- ReportLocation

Итерация 2:
- ModifyMap
- ModifyPoint

Итерация 3:
- PerformTask

Определим также набор классов, разработка которых приходится на указанные итерации. Отметим, что так как в качестве средства уменьшения риска разработки программной системы определено достижение работоспособности системы после каждой итерации, разработка классов ведется также целиком в рамках отведенной под эти цели итерации.

Итерация 0: (подготовительная итерация)

Разработка абстрактных классов CDevice, CControllerDevice и класса-шасси CMobileRobot

Итерация 1:
- Общие для разрабатываемых на данной итерации Use-case'ов классы:
  CRemoteInterface.
- Use-case: EmergencyStop
  Классы: CEmStopModule, CReactorGroup
- Use-case: ReportStatus
  Классы: CIntStatusModule
- Use-case: ReportLocation
  Классы: CLocationModule

Итерация 2:
- Общие для разрабатываемых на данной итерации Use-case'ов классы:
  CControlPanel, CMappingModule, CLocPoint
- Use-case: ModifyMap
  Классы: нет
- Use-case: ModifyPoint

Классы: нет

Итерация 3:

- Use-case: PerformTask

  Классы: CNavLogicUnit.

Дополнительно к вышеизложенным соображениям, приведем предварительную таблицу значений метрик, характеризующих качество разрабатываемой программной системы:

Метрика	CRemoteInterface	CLocationModule	CMappingModule	CNavLogicUnit	CIntStatusModule	CDevice	CEmStopModule	CReactorGroup	CControllerDevice	CControlPanel	CLocPoint	CMobileRobot	Среднее значение
WMC	1	1	5	5	1	1	1	1	1	0	0	0	1.41
NOC	0	0	0	0	0	7	0	0	2	0	0	0	0.75
Метрики, вычисляемые для системы													
DIT	3												
NC	12												
NOM	17 (28)												

Этап Конструирование.

Итерация 0: (подготовительная итерация)

Разработка абстрактных классов CDevice, CControllerDevice и класса-шасси CMobileRobot.

По каждому из вышеуказанных классов проводятся работы по реализации соответствующих операций:

Класс: **CDevice**
Абстрактный класс, являющийся суперклассом по отношению ко всем классам, описывающим аппаратные устройства, входящие в состав системы управления. Данный класс осуществляет включение и выключение описываемого устройства.
Класс обладает следующими методами:
Activate(bool State) – переводит устройство во включенное или выключенное состояние в зависимости от значения параметра State.

Класс: **CControllerDevice**
Абстрактный класс, являющийся суперклассом по отношению к классам CNavLogicUnit и CEmStopModule и реализующим функцию связи устройства с исполнительной группой посредством командной шины.
Класс обладает следующими методами:
RegisterReactorGroup(CReactorGroup* ReactorGroup) – ставит в соответствие данному экземпляру класса-источника логических команд экземпляр исполнительной группы контроллеров.

Класс: **CMobileRobot**
Данный класс описывает шасси, несущее устройства системы управления, и предназначен для хранения экземпляров классов-устройств системы управления.
Класс не обладает методами.

Для оценки качества проведенной разработки выполним подсчет значения метрик для реализованных классов и системы в целом:

Метрика	CDevice	CControllerDevice	CMobileRobot	Среднее значение
WMC	1	1	0	
NOC	7	2	0	
CBO	0	1	8	
RFC	0	0	0	
LCOM	0	0	0	
CS (a/o)	1/1	2/2	8/0	
NOO	0	0	0	
NOA	1	1	0	
SI	0	0	0	
OS_{avg}	0	0	0	
NP_{avg}	1	1	0	
Метрики, вычисляемые для системы				
DIT	2			
NC	3			
NOM	2			
LOC_{Σ}	-			

Итерация 1:

Реализация сценариев Use-case'ов:
- EmergencyStop
- ReportStatus
- ReportLocation

Как видно из диаграммы последовательности для данного Use-case'а, в ходе данной итерации необходимо реализовать следующие классы:

- CRemoteInterface
- CEmStopModule
- CReactorGroup
- CIntStatusModule
- CLocationModule

По каждому из вышеуказанных классов проводятся работы по реализации соответствующих операций:

Класс: **CRemoteInterface**
Данный класс описывает устройство *Интерфейс удаленного доступа* (на диаграммах последовательности – Remote Interface) и предназначен для обеспечения взаимодействия с диспетчером, производящим инициирование выполнения маршрутного задания и прочие действия.
Класс обладает следующими методами:
ConfirmStop() – посылает диспетчеру подтверждение о совершении аварийного останова через канал радиосвязи.

Класс: **CReactorGroup**
Данный класс описывает устройство *Исполнительная группа* (на диаграммах последовательности – Reactor Group) и предназначен для обеспечения взаимодействия между устройствами-источниками логических команд (модуль навигации и модуль аварийного останова) с аппаратными портами контроллеров приводов робота.
Класс обладает следующими методами:

SendCommand(int CmdID) – принимает посланную устройством-источником логическую команду, дешифрует и преобразует ее в набор аппаратно-зависимых команд и отправляет полученный набор в аппаратные порты контроллеров.

Класс: **CEmStopModule**
Данный класс описывает устройство *Модуль аварийного останова* (на диаграммах последовательности – Emergency Stop Module) и предназначен для посылки команды аварийного останова исполнительной группе по запросу, исходящему от модуля навигации, пульта управления или интерфейса удаленного доступа.
Класс обладает следующими методами:
Stop() – осуществляет посылку логической команды на аварийный останов исполнительной группе.

Класс: **CIntStatusModule**
Данный класс описывает устройство *Модуль проверки состояния* (на диаграммах последовательности – Internal Status Module) и предназначен для сбора и выдачи информации о состоянии аппаратных подсистем робота с целью обеспечения возможности принятия решения о возможности или невозможности дальнейшего движения.
Класс обладает следующими методами:
GetStatus() – производит опрос аппаратных подсистем робота и выдает сообщение о наличии или отсутствии целостности в проверяемых подсистемах.

Класс: **CLocationModule**
Данный класс описывает устройство *Модуль определения места расположения* (на диаграммах последовательности – Location Module) и предназначен для выдачи по запросу интерфейса удаленного доступа или модуля навигации текущих координат робота.
Класс обладает следующими методами:
GetLocation() – производит считывание текущих координат робота с аппаратных портов, взаимодействующих через контроллеры с датчиками положения и возвращает полученный результат.

Для оценки качества проведенной разработки выполним подсчет значения метрик для реализованных классов и системы в целом:

Метрика	CRemoteInterface	CLocationModule	CIntStatusModule	CDevice	CEmStopModule	CReactorGroup	CControllerDevice	CLocPoint	CMobileRobot	Среднее значение
WMC	1	1	1	1	1	1	1	0	0	
NOC	0	0	0	7	0	0	2	0	0	
CBO	4	0	0	0	3	0	1	0	8	
RFC	0	0	0	0	3	0	0	0	0	
LCOM	0	0	0	0	0	0	0	0	0	
CS (a/o)	4/2	1/2	1/2	1/1	4/3	1/2	2/2	2/0	8/0	
NOO	0	0	0	0	0	0	0	0	0	
NOA	1	1	1	1	1	1	1	0	0	
SI	0	0	0	0	0	0	0	0	0	
OS_{avg}	0	0	0	0	3	0	0	0	0	
NP_{avg}	0	0	0	1	0	1	1	0	0	
Метрики, вычисляемые для системы										
DIT	3									
NC	9									
NOM	7									
$LOC_{\sum}$										

Итерация 2:

Реализация сценариев Use-case'ов:
- ModifyMap
- ModifyPoint

Как видно из диаграммы последовательности для данного Use-case'а, в ходе данной итерации необходимо реализовать следующие классы:

- CControlPanel
- CMappingModule

По каждому из вышеуказанных классов проводятся работы по реализации соответствующих операций:

Класс **CControlPanel**
Данный класс описывает устройство *Пульт управления* (на диаграммах последовательности – Control Panel) и предназначен для осуществления взаимодействия с оператором, выполняющим функции изменения карты местности и программирования целевых точек.
Класс обладает следующими методами:
Нет методов.

Класс **CMappingModule**
Данный класс описывает устройство *Модуль хранения карты* (на диаграммах последовательности – Mapping Module) и предназначен для хранения и обеспечения доступа к карте местности. Данный класс также предназначен для хранения координат программируемых целевых точек.
Класс обладает следующими методами:
SetDim(CLocPoint Dimensions) – устанавливает размеры хранимой карты местности.
SetMapEl(CLocPoint Location, int Element) – устанавливает значение указанного элемента карты местности.
SetPoint(int index, CLocPoint Location) – устанавливает координаты указанной программируемой целевой точки.
GetPointLoc(int index) – Возвращает координаты указанной программируемой целевой точки.

GetMapData(CLocPoint Location) – возвращает значение указанного элемента карты местности.

Для оценки качества проведенной разработки выполним подсчет значения метрик для реализованных классов и системы в целом:

Метрика	CRemoteInterface	CLocationModule	CMappingModule	CIntStatusModule	CDevice	CEmStopModule	CReactorGroup	CControllerDevice	CControlPanel	CLocPoint	CMobileRobot	Среднее значение
WMC	1	1	5	1	1	1	1	1	0	0	0	
NOC	0	0	0	0	7	0	0	2	0	0	0	
CBO	4	0	1	0	0	3	0	1	2	0	8	
RFC	0	0	0	0	0	3	0	0	0	0	0	
LCOM	0	0	0	0	0	0	0	0	0	0	0	
CS (a/o)	4/2	1/2	4/6	1/2	1/1	4/3	1/2	2/2	3/1	2/0	8/0	
NOO	0	0	0	0	0	0	0	0	0	0	0	
NOA	1	1	5	1	1	1	1	1	0	0	0	
SI	0	0	0	0	0	0	0	0	0	0	0	
OS_{avg}	0	0	0	0	0	3	0	0	0	0	0	
NP_{avg}	0	0	1.6	0	1	0	1	1	0	0	0	
Метрики, вычисляемые для системы												
DIT	3											
NC	11											
NOM	12											
LOC_{Σ}												

Итерация 3:

Реализация сценариев Use-case'ов:

- PerformTask

Как видно из диаграммы последовательности для данного Use-case'а, в ходе данной итерации необходимо реализовать следующие классы:

- CNavLogicUnit

По каждому из вышеуказанных классов проводятся работы по реализации соответствующих операций:

Класс **CNavLogicUnit**

Данный класс описывает устройство *Модуль навигации* (на диаграммах последовательности – Navigation Logic Unit) и предназначен для осуществления управлением роботом через исполнительную группу согласно вычисляемому на каждом такте работы модуля маршрутному заданию.

Класс обладает следующими методами:

ResetPoint() – производит сброс текущей выбранной точки и текущего маршрутного задания.

SelectPoint(int point) – производит выбор новой целевой точки по ее индексу размещения в памяти модуля хранения карты.

Start() – производит запуск процесса пошагового продвижения робота к выбранной целевой точке.

Calc() – производит пересчет текущего маршрутного задания сообразно с текущим местом расположения робота, координатами выбранной целевой точки и содержимым карты местности.

ValidateLocation() – производит сверку фактического места расположения с расчетным после выполнения очередного шага маршрутного задания.

Для оценки качества проведенной разработки выполним подсчет значения метрик для реализованных классов и системы в целом:

Метрика	CRemoteInterface	CLocationModule	CMappingModule	CNavLogicUnit	CIntStatusModule	CDevice	CEmStopModule	CReactorGroup	CControllerDevice	CControlPanel	CLocPoint	CMobileRobot	Среднее значение
WMC	1	1	5	5	1	1	1	1	1	0	0	0	1.41
NOC	0	0	0	0	0	7	0	0	2	0	0	0	0.75
CBO	4	0	1	6	0	0	3	0	1	2	0	8	2.08
RFC	0	0	0	8	0	0	3	0	0	0	0	0	0.91
LCOM	0	0	0	0	0	0	0	0	0	0	0	0	0
CS (a/o)	4/2	1/2	4/6	8/7	1/2	1/1	4/3	1/2	2/2	3/1	2/0	8/0	3.25/2.33
NOO	0	0	0	0	0	0	0	0	0	0	0	0	0
NOA	1	1	5	5	1	1	1	1	1	0	0	0	1.41
SI	0	0	0	0	0	0	0	0	0	0	0	0	0
OS_{avg}	0	0	0	2.4	0	0	3	0	0	0	0	0	0.45
NP_{avg}	0	0	1.6	0.2	0	1	0	1	1	0	0	0	0.4
Метрики, вычисляемые для системы													
DIT	3												
NC	12												
NOM	17												
$LOC_{\sum}$													

Конечная диаграмма классов.

В результате произведенных итераций конструирования программной системы, была построена система классов, реализующая все требуемые механизмы взаимодействия между узлами системы управления. Конечная диаграмма классов, отображающая структуру системы классов, приведена на рис. 22, 23, 24.

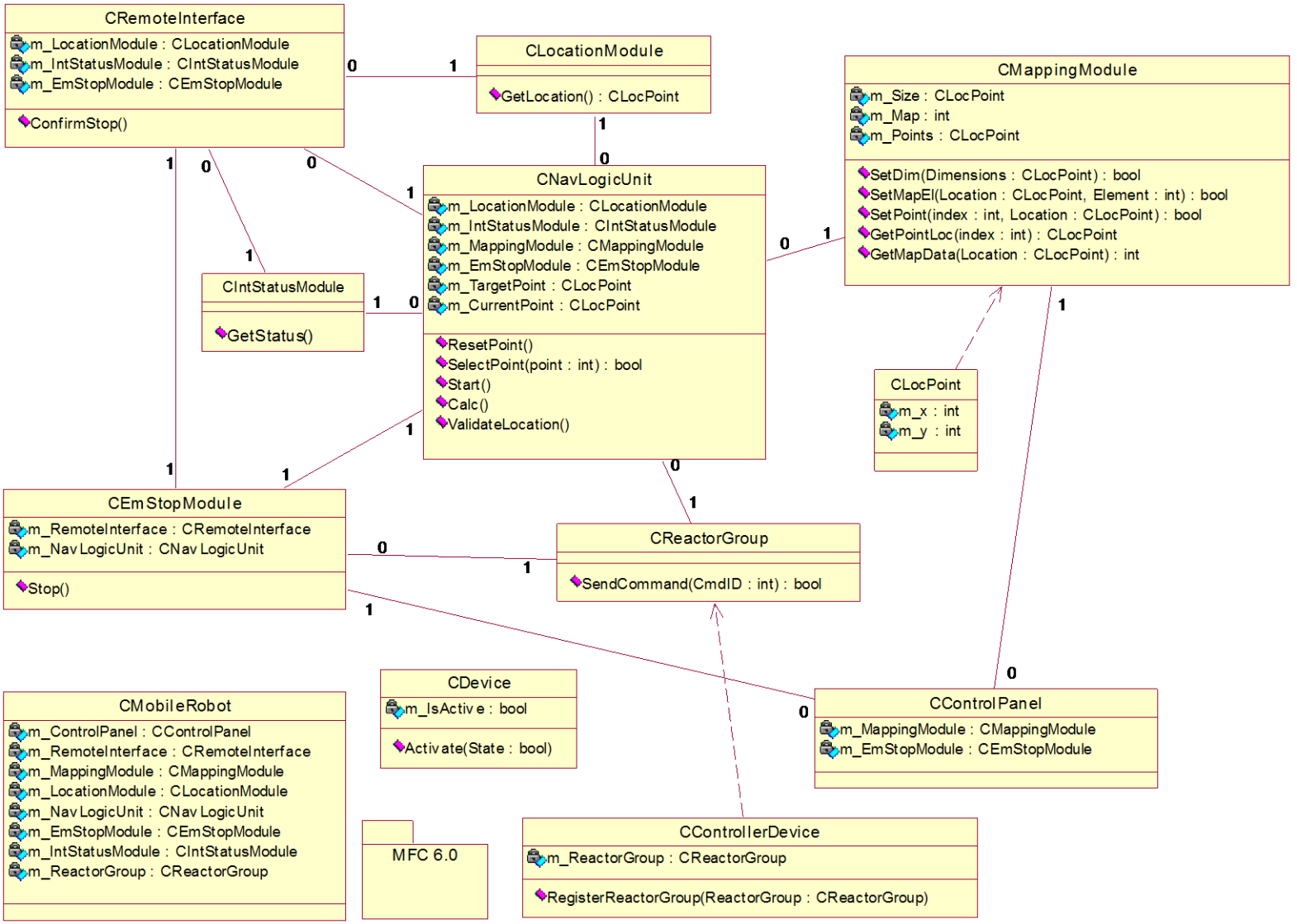

Рис. 22. Конечная диаграмма классов. Отношения ассоциации и зависимости

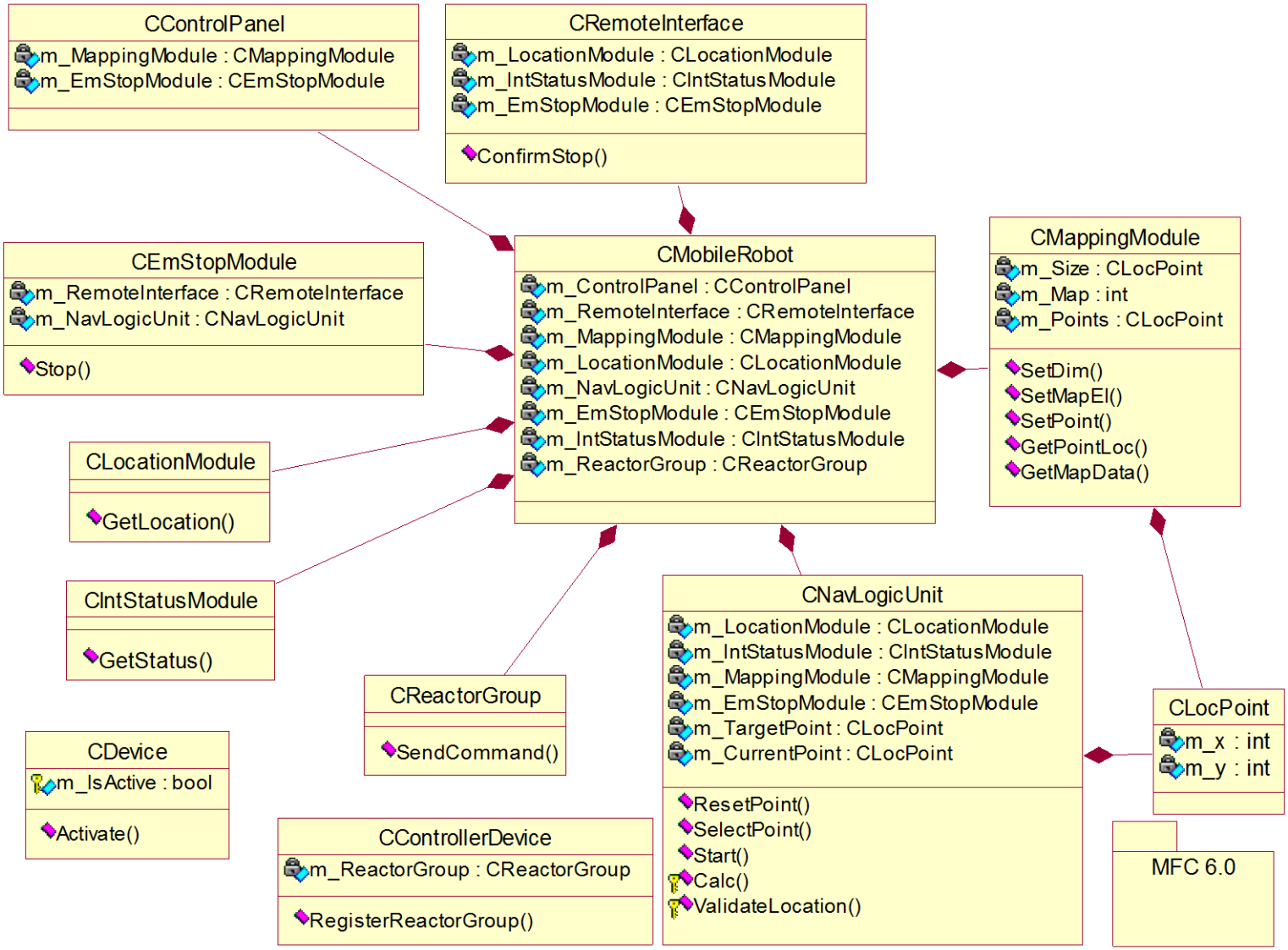

Рис. 23. Конечная диаграмма классов. Отношения агрегации.

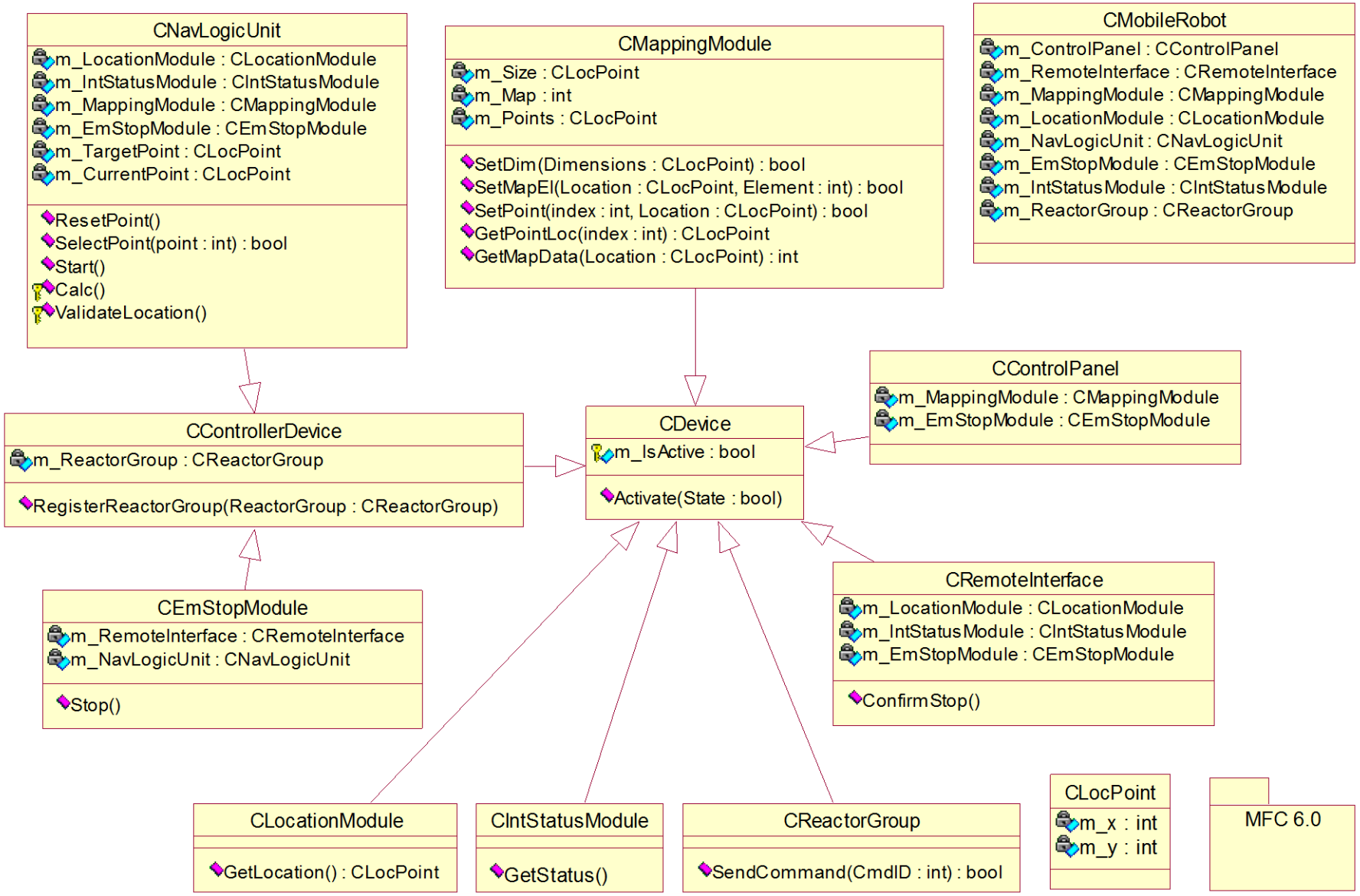

Рис. 24. Конечная диаграмма классов. Отношения обобщения.

В дополнение к вычисляемым на каждой итерации метрик Чидамбера и Кемерера (WMC, DIT, NOC, CBO, RFC, LCOM), а также Лоренца и Кидда (CS, NOO, NOA, SI, OSavg, NPavg), предназначенных для оценки качества разработки объектно-ориентированных программных систем, вычислим значения набора интегральных метрик Фернандо Абреу для всего проекта в целом.

Метрика	Значение
MHF	2/17
AHF	1
MIF	11/29
AIF	11/39
POF	0
COF	7/33

Выводы.

В результате проделанной работы была получена объектно-ориентированная программная система с заданными характеристиками и отвечающая поставленным требованиям. По существу процесса создания программной системы можно сделать следующие замечания:

Общие выводы.

- Построение программной системы выполнялось в соответствии с разработанной схемой аппаратной реализации, благодаря чему разработанные программные классы в точности соответствуют аппаратным устройствам и выполняют возложенные на них функции. Такой подход наиболее приемлем для разработки программных систем, рассчитанных либо на установку в аппаратную среду, либо на моделирование системы аппаратных устройств. Также данный подход позволяет четко определить функции каждого из классов и уменьшить степень свободы на одном из наиболее критичных этапов разработки – этапе построения диаграмм последовательности. Степень свободы уменьшается благодаря повышению определенности относительно разрабатываемых на базе схемы аппаратной реализации классов, что ведет к

уменьшению степени риска при разработки и снижению вероятности появления и накопления ошибок и неверных проектных решений.

- В ходе разработки удалось применить механизм наследования классов, благодаря чему повысилась степень повторного использования, что ведет к повышению надежности системы и уменьшению плотности ошибок, допускаемых на этапе конструирования в ходе непосредственного написания кода.
- В ходе планирования итераций конструирования удалось найти решение, позволяющее получать после каждой завершенной итерации не только законченные с точки зрения программной реализации классы, конструирование которых планировалось на данной итерации, но и работоспособную (в рамках разрабатываемого Use-case'а) систему, что позволяет не только значительно уменьшить риск разработки посредством обеспечения возможности отладки кода на каждой итерации, но и снизить накладные расходы, связанные с достаточно трудоемким процессом пересчета метрик, в случае если разработка методов класса ведется в ходе нескольких итераций.

Выводы относительно значений метрик.

На основании вычисленных значений метрик качества программной системы можно сделать ряд выводов об адекватности принятых в ходе разработке решений и качестве конечного результата разработки.

- Значение метрики AHF (Attribute Hiding Factor), равное 1 свидетельствует о хорошем стиле объектно-ориентированного программирования, подразумевающего полное закрытие атрибутов класса от непосредственного внешнего доступа. Такая черта значительно повышает надежность разработанных классов и увеличивает простоту и однозначность их повторного использования.
- Было получено достаточно низкое значение метрики COF, что свидетельствует о малом сцеплении между объектами, что также снижает плотность дефектов и затрат на доработку.
- Значение метрики DIT, равное трем свидетельствует о том, что механизм наследования с одной стороны присутствует в системе, а с другой стороны система использует этот механизм ровно в той мере, в которой это требуется исходя их специфики разрабатываемого продукта, без излишнего увеличения степени сложности дерева наследования.
- Были получены низкие значения метрики LCOM, что свидетельствует о том, что методы достаточно хорошо связаны через свойства, что уменьшает сложность и вероятность ошибок. Обратный результат свидетельствовал бы о том, что классы являются не более чем набором разрозненных функций.
- Значение метрики CS не превысило рекомендуемые 20 методов.
- Значение метрики NOO не превысило рекомендуемые 3 метода, составив однако, нулевое значение. Данный факт свидетельствует о том, что в рамках данной работы не был использован механизм переопределения методов. Следует

отметить, что подобного рода недостаток объясняется спецификой разрабатываемой системы, которая состоит из в достаточной степени сильно диверсифицированных устройств, не имеющих возможности эффективно использовать механизм переопределения методов.

- Значение метрики NOA не превысило рекомендуемые 4 метода.
- Значение метрики OSavg не превысило рекомендуемые 9 сообщений.
- Среднее значение метрики NPavg составило значение 0.4 параметра на метод, что несколько отличается от рекомендуемого значения в 0.7. Однако заниженное значение метрики NPavg не свидетельствует о том, что сигнатуры методов спроектированы с избыточной степенью сложности, что способствует лучшему восприятию исходного кода.
- Значение метрики MIF не превысило рекомендуемый порог в 70%, что свидетельствует о том, что наследование на уровне методов используется в умеренной, но в тоже время достаточной форме.