**Пояснительная записка**

К программной системе «Система Обеспечения Такси»

По предмету «Технологии и разработки прикладных программных систем»

**Проект подготовили студенты группы Р05-281:**

*Бондаренко Юрий*

*Галаев Александр*

*Галиев Булат*

*Галкин Тимофей*

*Евтушенко Данила*

Разделы пояснительной записки:

1. [Общие положения](http://www.prj-exp.ru/patterns/pattern_tech_project.php#tech_project1)
   * Наименование системы
   * Цели, назначение и области использования системы
   * Нормативные ссылки
   * Очередность создания системы
2. [Основные технические решения](http://www.prj-exp.ru/patterns/pattern_tech_project.php#tech_project2)
   * [Сведения об обеспечении заданных потребительских характеристик системы, определяющих ее качество](http://www.prj-exp.ru/patterns/pattern_tech_project.php#tech_project25)
   * [Средства разработки](http://www.prj-exp.ru/patterns/pattern_tech_project.php#tech_project29)
3. Заключение

Пояснительная записка к техническому проекту на создание программной системы «Система обеспечения такси»

1. Общие положения

1.1. Наименование системы

1.1.1. Полное наименование системы

Полное наименование – Система обеспечения такси.

1.1.2. Краткое наименование системы

Краткое наименование - СОТ, Система.

1.2. Основания для проведения работ

Работа выполняется на основании договора между НИЯУ МИФИ и Студентами, в рамках которого последние обязаны изучить список вверенных дисциплин.

1.3. Наименование организаций – Заказчика и Разработчика

1.3.1. Заказчик

Заказчик:

Сергиевский.М.В. ...

1.3.2. Разработчик

Разработчик:

Студенты Р05-281: Бондаренко Ю., Галкин Т., Галаев А., Галиев Б., Евтушенко.Д.

1.4 Цели, назначение и область использования системы

Цель: Моделирование части городской инфраструктуры на примере организации работы такси.

Назначение: Усвоение Студентами программы курса «Технологии и разработки прикладных программных систем», а также приобретение в ходе работы необходимых теоретических и практических навыков.

Область использования: система может быть использована в ходе обучающего процесса.

1.5. Нормативные ссылки

При техническом проектировании использовались следующие нормативно-технические документы:

Гост 19.404-79

1.6 Очередность создания системы

**Итерация 1**

**Анализ требований**

Была сформулирована расширенная постановка задачи:

Имеется M машин такси, заранее расположенных по городу, и L пассажиров.

Число мест в каждом такси - 4. Пассажиры располагаются в заранее определённых местах посадки; программа работает до тех пор, пока система такси не развезёт всех пассажиров.

Такси может подсаживать пассажиров, если это не приводит к изменению маршрута. Такси, высадившее пассажиров, остаётся на пункте, пока не поступит новый заказ.

Задача моделируется, опираясь на тот факт, что в «городе» отсутствуют пробки и светофоры, задерживающие такси. (Такси всё время движется с постоянной скоростью).

Было решено, что диаграмма прецедентов должна содержать следующие функции:

**Для пассажира такси:**

-Вызвать ближайшее такси

-Установить конечную точку

**Для такси:**

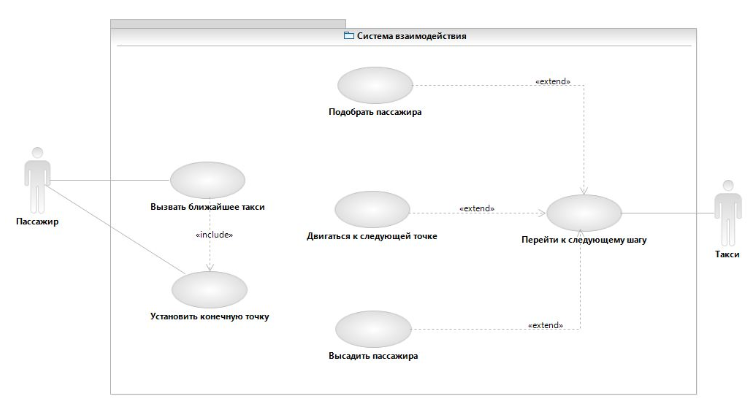
-Перейти к следующему шагу

-Подобрать пассажира

-Двигаться к следующей точке

-Высадить пассажира

На основе уточнённой задачи была построена диаграмма прецедентов.



**Проектирование**

Было решено, что на первой итерации должно существовать 3 класса, а именно:

-Такси, в котором содержится информация о позиции, маршруте, вместимости и скорости такси. Данный класс содержит следующие поля:

* Позиция
* Маршрут
* Вместимость
* Скорость

А также методы:

* Добавить пассажира
* Действие
* Высадить пассажира

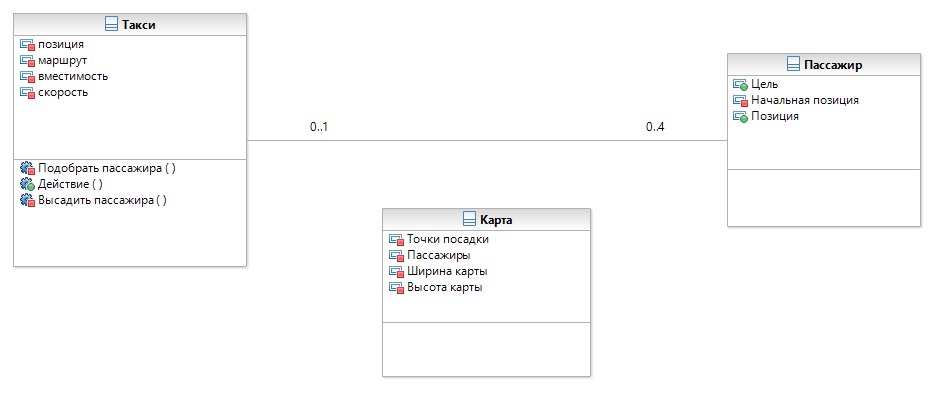
-Пассажир, в котором содержится информация о его позиции, пункте назначения и текущем местоположении. Класс Пассажир содержит следующие поля:

* Цель
* Начальная позиция
* Позиция

-Карта, состоящая из точек посадки (пункты и связи между ними), и включающая в себя список пассажиров. Класс карта содержит следующие поля:

* Точка посадки
* Пассажиры
* Ширина карты
* Высота карты

Для выполнения поставленной задачи была предложена следующая диаграмма классов



Данная диаграмма содержит следующие классы: Такси, Пассажир, Карта;

Между классами Такси и Пассажир введено отношение ассоциации, кратность которого показывает, что в одном такси может ехать до 4 пассажиров. Тем не менее, такси так же может не везти пассажиров вовсе.

На текущем уровне абстракции класс Карта напрямую не взаимодействует ни с одним из уже существующих классов в силу отсутствия необходимости подобных действий.

**Программирование**

Основной задачей стадии программирования на первой итерации было построение первичной общей структуры кода, соответственной текущему уровню абстракции.

Хоть эта структура и не в полной мере отражает функциональность, ожидаемую заказчиком, она создаёт первоначальный фундамент для дальнейшего развития и улучшения программной системы, наращивания более полного функционала.

**Итог**

#### На первой итерации обсуждалась общая структура проекта, и был произведен анализ предметной области. Был определён начальный состав классов и спроектирована система отношений между ними.

#### Данная версия программной системы пока что не предусматривает отображение какого-либо интерфейса. Особое внимание на первой итерации было уделено отношениям между классами.

**Итерация 2**

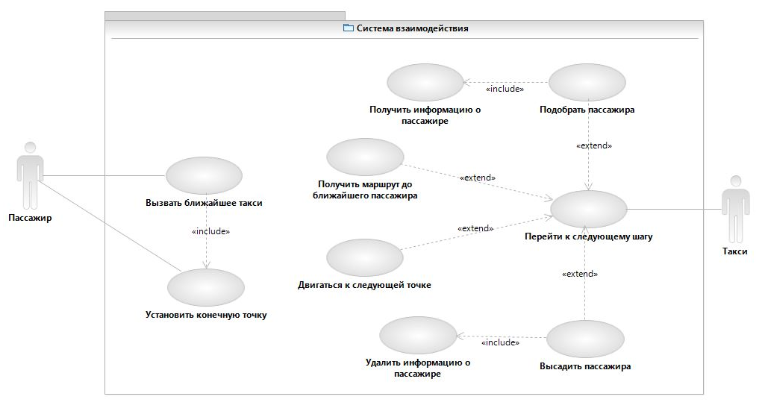
**Анализ требований**

Основные цели:

* Улучшить функциональность кода путём использования шаблонов проектирования
* Добавить необходимые методы
* Ввести требуемые ассоциативные связи
* Реализовать базовый интерфейс

**Диаграмма прецедентов**

При использовании шаблона проектирования Controller была введена новая сущность, ответственная за хранение данных и взаимодействие между классами. Поэтому исходная диаграмма прецедентов не претерпела каких-либо изменений.



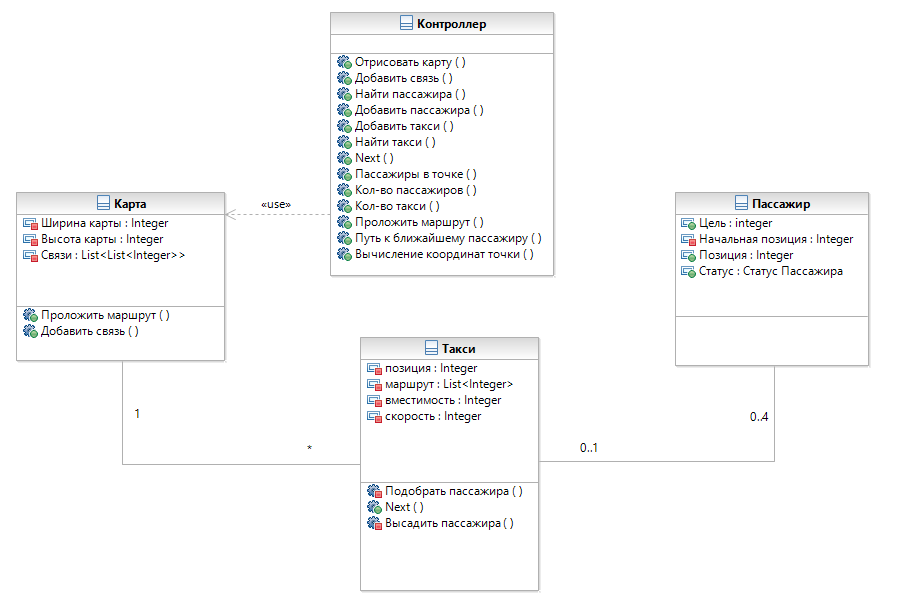
**Проектирование**

После решения о вводе дополнительного класса «Контроллер», диаграмма классов, представленная в первой итерации, была дополнена.

Класс «Контроллер» был создан для реализации хранения данных и взаимодействия между тремя классами, представленными в первой итерации.

Было решено, что данный класс должен содержать следующие методы, которое будут присущи ему на данной итерации:

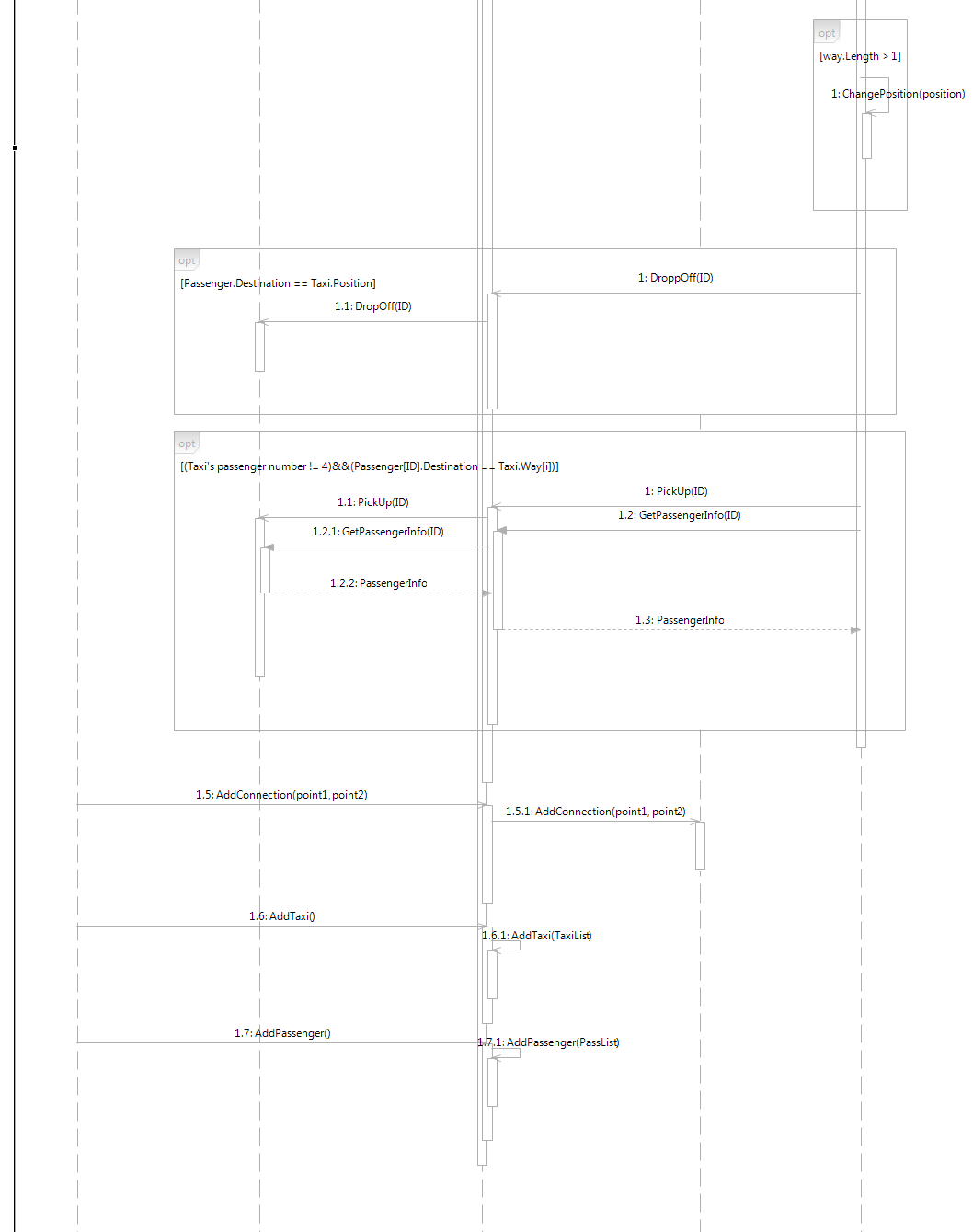
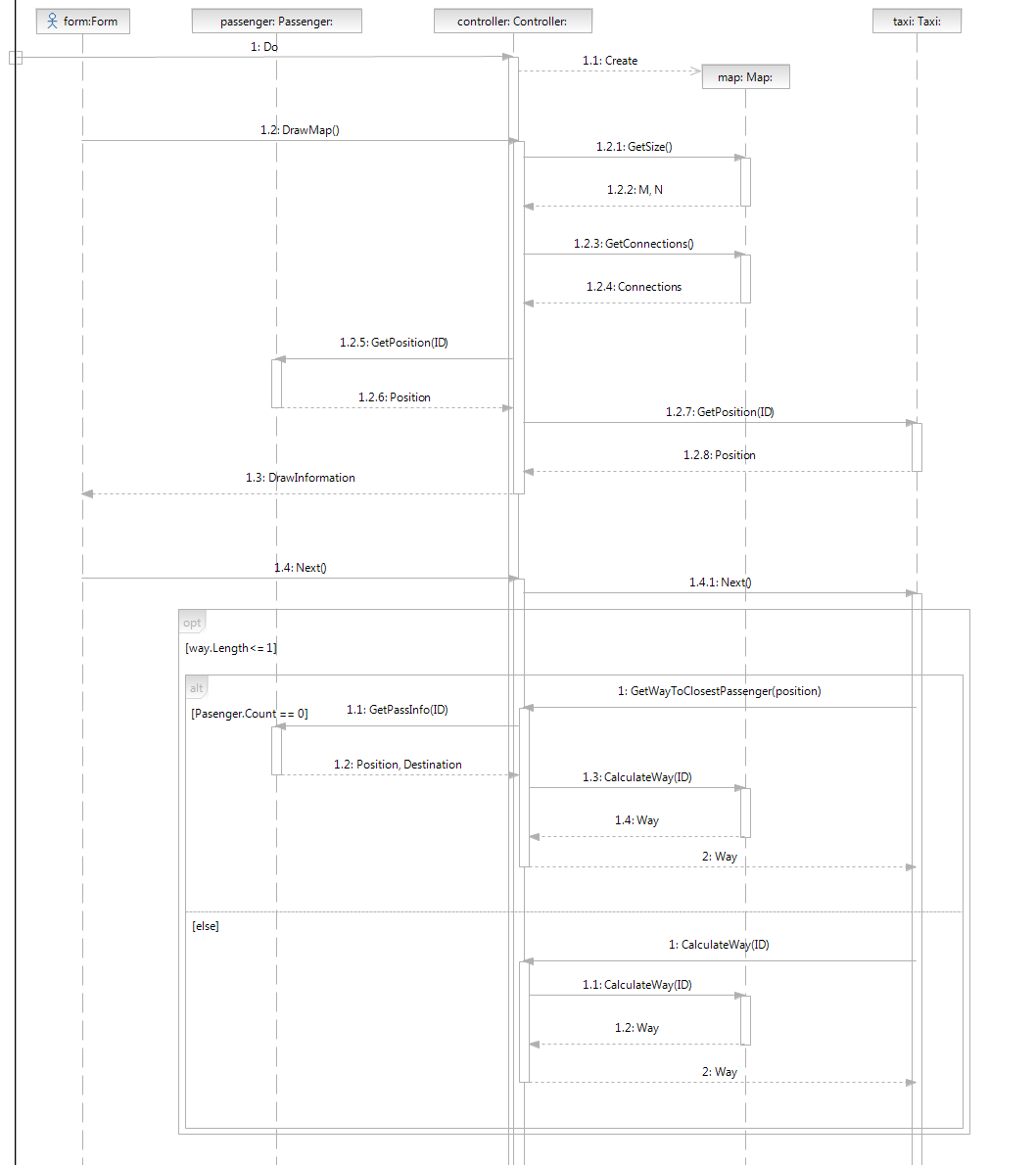
* Отрисовывать карту
* Добавить связь к точке
* Найти пассажира
* Добавить пассажира в список
* Добавить такси в список
* Найти такси
* Next() – отвечает за переход к следующему шагу на карте



Класс Контроллер и класс Карта связаны отношением зависимости.

Класс Карта и класс Такси теперь связаны отношением ассоциации типа «один ко многим», что отражает факт того, что одному объекту класса Карта может быть присущ некоторый набор объектов класса Такси.

Чтобы более детально описать упорядоченное по времени взаимодействие объектов с последовательностью их появления, была построена диаграмма последовательностей для прототипа, планируемого на данном итерационном шаге разработки программной системы.

****

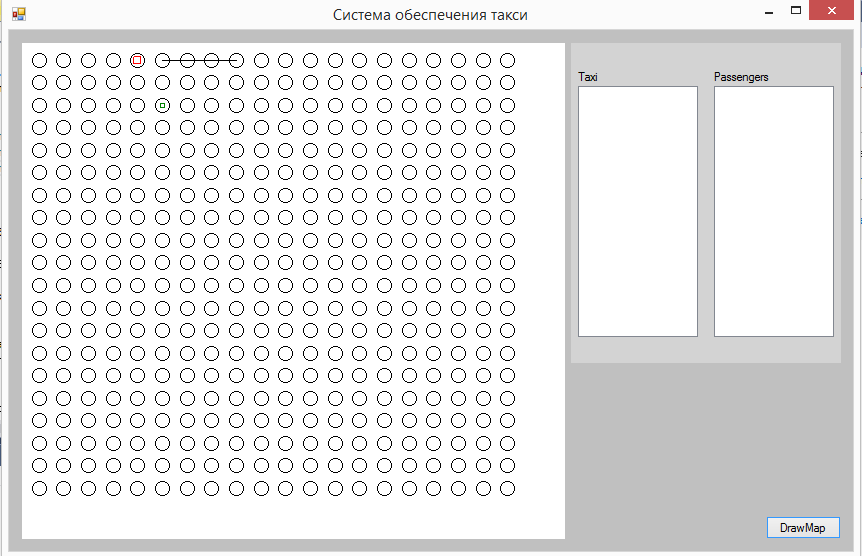
**Программирование**

Была визуализирована первичная модель уличной системы. Каждый круг на форме символизирует собой остановочный пункт;

Красный квадрат – одна из машин такси, осуществляющих функции транспортировки пассажиров.

Зеленый квадрат – ожидающий пассажир.

Линия – связь (дорога), соединяющая остановочные пункты.



**Итог**

Во второй итерации был использован шаблон проектирования Controller для того, чтобы осуществлять грамотное взаимодействие между различными классами системы.

Появилась связь между классом Карта и введённым классом Контроллер, а также между классом Карта и классом Такси.

Был реализован первоначальный вариант интерфейса для программной системы.

**Итерация 3**

**Анализ требований**

Основные цели:

* Добавление новых методов в классы и, как следствие, уменьшение уровня абстракции и наращивание функциональности системы.
* Введение новых классов.
* Добавление новой вспомогательной структуры данных для упрощения читаемости кода. (Новый enum PassengerStatus)
* Доработка существующего интерфейса.

Диаграмма прецедентов

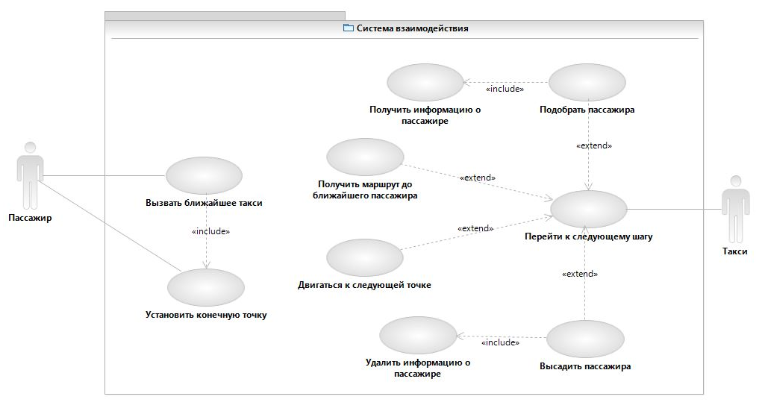
Было решено дополнительно ввести следующий набор функций, оптимизирующих работу программы:

-Удалить информацию о пассажире

-Получить информацию о пассажире

-Получить маршрут до ближайшего пассажира

Новая диаграмма прецедентов имеет следующий вид:



**Проектирование**

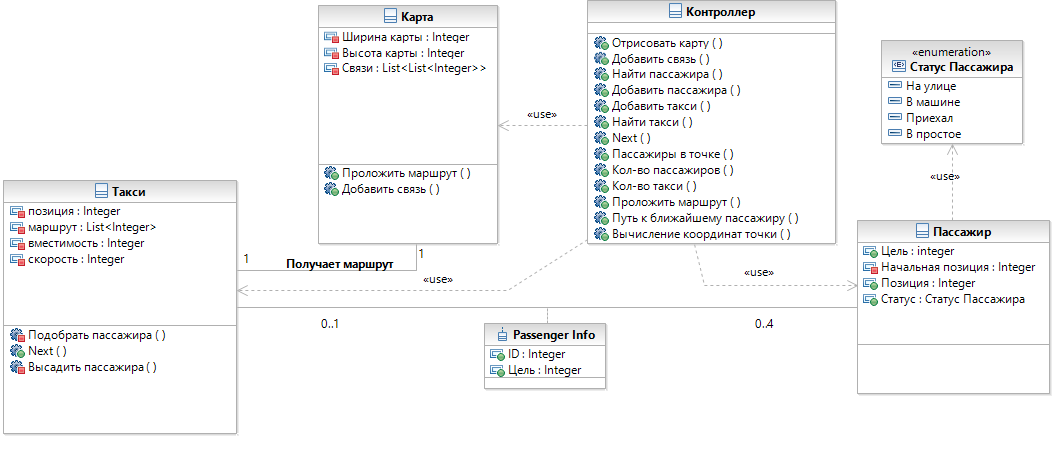
Введение класса-ассоциации и перечисления для улучшения качества кода с точки зрения читаемости.

Класс-ассоциация PassengerInfo содержит следующие поля:

* ID – номер пассажира в массиве Controller.Passengers
* Цель – номер пункта назначения в массиве Controller.Map.Connections

Перечисление (новый enum PassengerStatus) будет отвечать за текущий статус пассажира, и уточнять следующую информацию:

* Находится ли пассажир на улице (или в машине)
* Добрался ли пассажир до своей цели.
* Находится ли пассажир в ожидании вызванной машины в настоящий момент времени.



Класс Контроллер и Такси связаны между собой связью зависимости.

Класс Пассажир так же соединен с классом Контроллер связью типа зависимость.

Класс Карта соединён с классом Контроллер связью типа зависимость.

Список изменений в классах:

Добавлены новые методы в класс Контроллер

* Пассажиры в точке
* Кол-во пассажиров
* Кол-во такси
* Проложить маршрут
* Путь к ближайшему пассажиру
* Вычисление координат точки

Добавлены новые методы в класс Карта

* Проложить маршрут
* Добавить связь

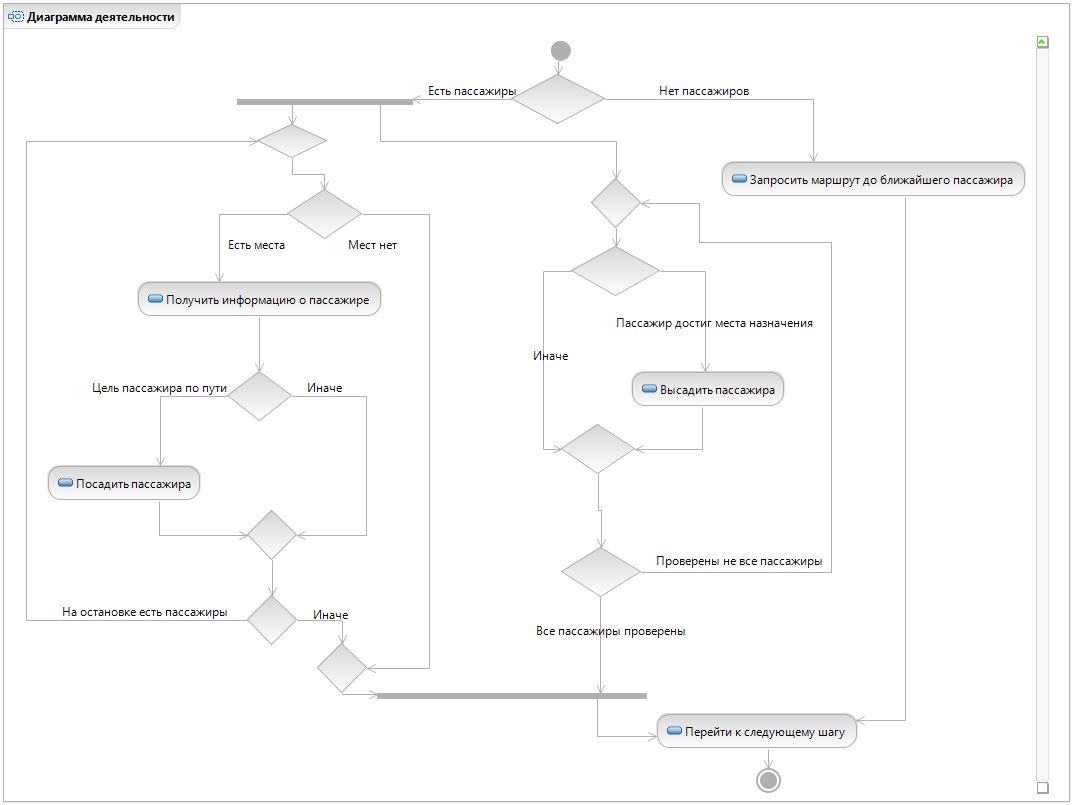
Так же, для улучшения эффективности и читаемости кода было принято решение о введении вспомогательной структуры данных под названием enumeration, отвечающей за более детальное описание текущего состояния пассажира.

Был введён класс-ассоциация PassengerInfo для описания ассоциативной связи, введение которой привело к увеличению эффективности кода.

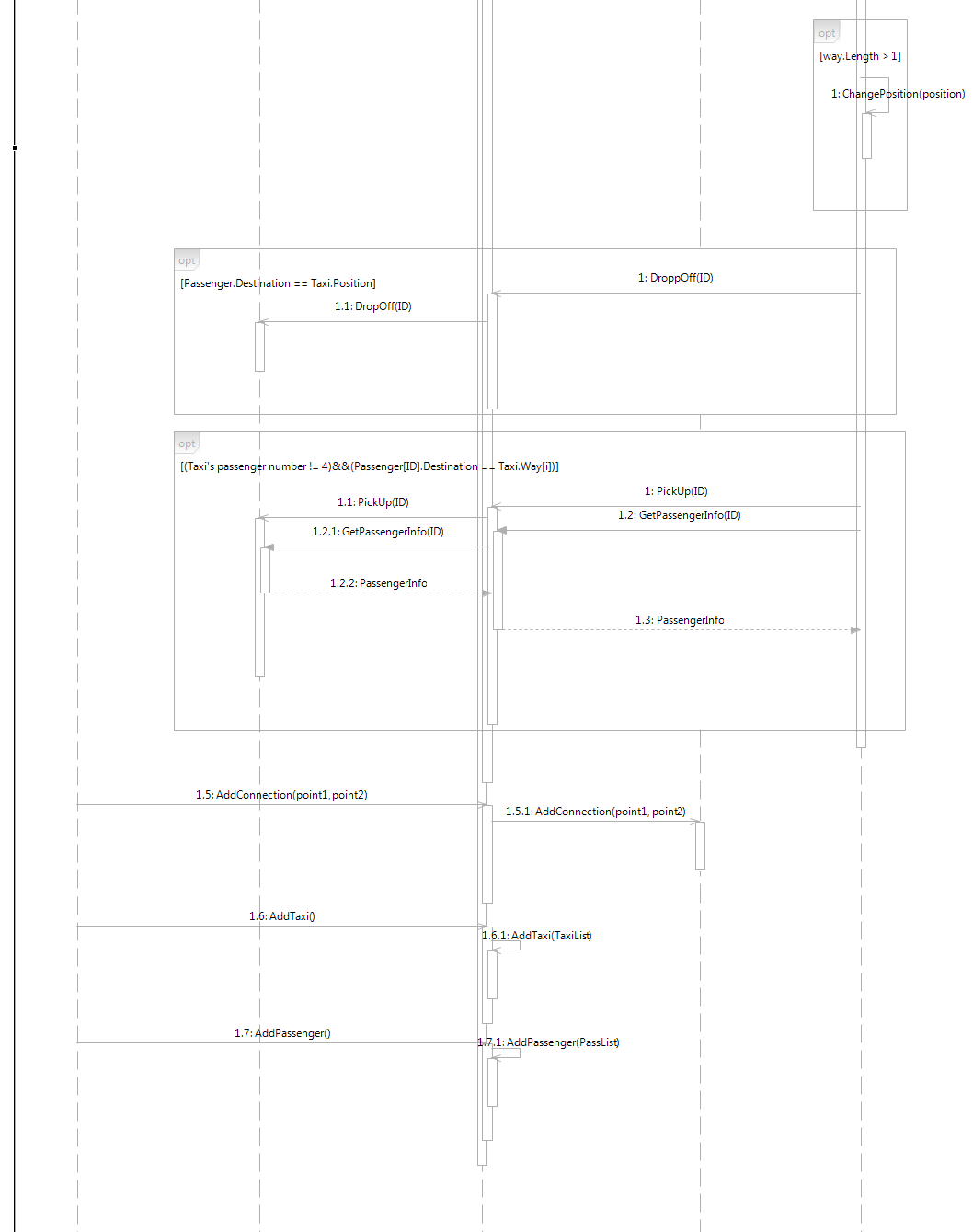
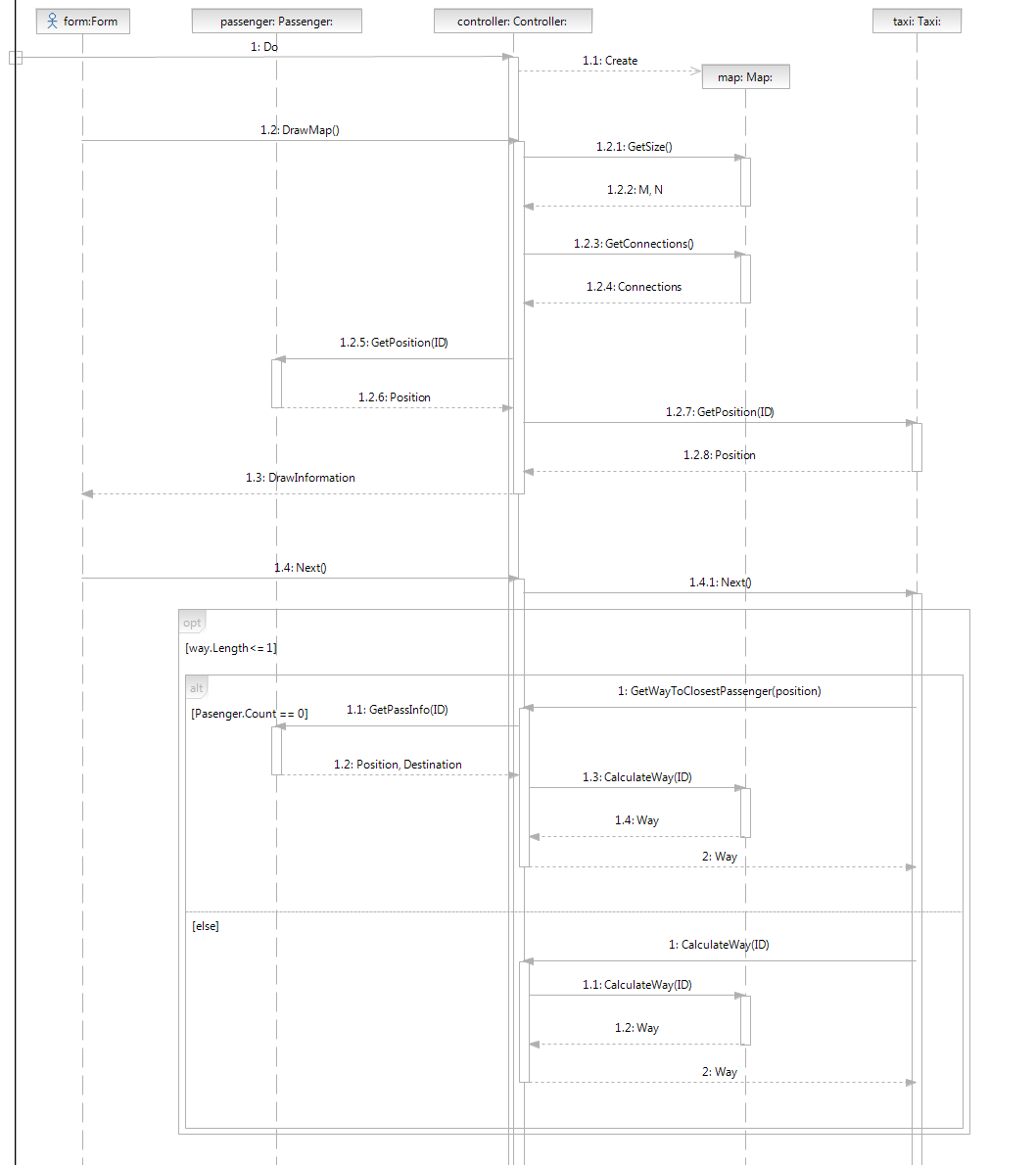
Далее необходимо представить структурную схему, служащую для описания алгоритма работы программной системы

Прибегнем для этого к диаграмме деятельности.

Цикл в диаграмме необходим для того, чтобы такси имело возможность забирать по несколько пассажиров с одной остановки при условии, что данный пассажир не нарушит заранее намеченного пути следования транспорта (для этого предусмотрен отдельный блок).



При дорабатывании кода не понесла изменений основная структура взаимодействий, поэтому диаграмма последовательностей не изменила своего первоначального вида.



**Программирование**

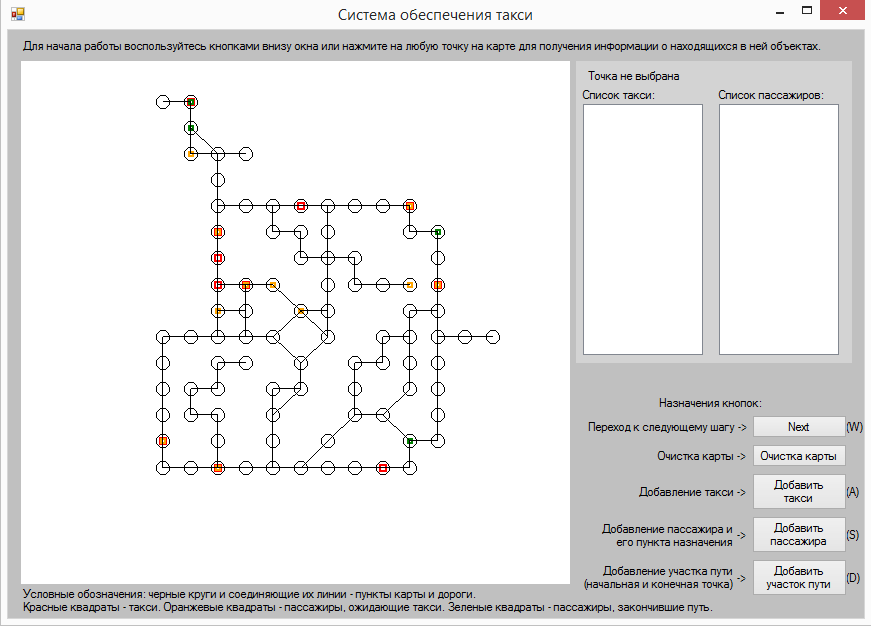
Реализация класса-ассоциации и перечисления для улучшения качества кода с точки зрения читаемости. Так же был доработан интерфейс системы:

Красным выделяются такси

Оранжевым выделяются пассажиры, ожидающие такси

Зелёными прямоугольниками обозначаются пассажиры на остановочных пунктах, уже добравшиеся до своей цели

Линиями отображаются существующие связи между пунктами



* Отображаются только уже введённые остановочные пункты и связи между ними
* Введена возможность добавления пассажиров на карту «на лету»
* Введена возможность добавления такси на карту «на лету»
* Улучшена функция поиска пути
* Добавлена функция сравнения пути, не позволяющая взять в такси пассажира, чей маршрут следования нарушает маршрут такси.
* В целях улучшения дружелюбности интерфейса добавлены пояснения рядом с кнопками, а также вверху и внизу формы.
* Добавлены функционирующие информационные окна в правом верхнем углу формы, отвечающие за статус пассажиров и такси.

**Тестирование и отладка**

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Реальный результат |
| Добавление новой связи на модель городской структуры | Создаётся новая связь. Если связь в проводится в место, где до этого не было остановочного пункта, он там появляется. | Создаётся новая связь. Если связь в проводится в место, где до этого не было остановочного пункта, он там появляется. |
| Добавление нового пассажира | По первому клику на карте задаётся позиция нового пассажира, по второму место его назначения. Если на первом или втором этапе пользователь указал место, где не существует остановочного пункта, то система предупредит его об этом (попросит указать корректные данные) | По первому клику на карте задаётся позиция нового пассажира, по второму место его назначения. Если на первом или втором этапе пользователь указал место, где не существует остановочного пункта, то система предупредит его об этом (попросит указать корректные данные) |
| Добавление такси | Указывается остановочная точка для добавления новой машины такси. Если пользователь указал место, где не существует остановочного пункта, то система предупредит его об этом (попросит указать корректные данные) | Указывается остановочная точка для добавления новой машины такси. Если пользователь указал место, где не существует остановочного пункта, то система предупредит его об этом (попросит указать корректные данные) |
| Очистка карты | Происходит полное стирание всей уличной структуры, включая пункты, связи между ними, пассажиров и такси. | Происходит полное стирание всей уличной структуры, включая пункты, связи между ними, пассажиров и такси. |
| Переход к следующему шагу | Машины такси, при наличии заказа, едут в указанную точку. При отсутствии такового просто стоят на месте в ожидании. | Машины такси, при наличии заказа, едут в указанную точку. При отсутствии такового просто стоят на месте в ожидании. |
| Клик на случайной ячейке | В правом верхнем углу в соответствующих окнах отображается информация об объектах, находящихся в данной ячейке. | В правом верхнем углу в соответствующих окнах отображается информация об объектах, находящихся в данной ячейке. |

**Итог**

Итогом работы на данной итерации является расширение интерфейса, функциональности. Внесённые изменения позволяют более наглядно выразить суть и назначение программной системы.

Работа нашей группы привела к созданию продукта, позволяющего смоделировать часть городской инфраструктуры на примере работы службы такси.

**2. Основные технические решения**

2.1. Сведения об обеспечении заданных в техническом задании потребительских характеристик системы, определяющих ее качество

|  |  |
| --- | --- |
| **Требование** | **Подход** |
| Функциональность | Углубленный анализ требований заказчика, взаимодействие с ним на каждом этапе разработки программной системы. |
| Эффективность | Оптимизация объёмов используемых ресурсов путём введения и последующего использования динамических структур данных. |
| Практичность | Лёгкость эксплуатации программного обеспечения достигается благодаря узкой группировке и широкой взаимосвязанности реализуемых функций. |
| ... | ... |

**2.2. Средства разработки**

Для создания ПС использовалось программное обеспечение, включающее IBM Rational Software Architect, интегрированную среду разработки Microsoft Visual Studio. Для работы в Visual Studio использовался язык программирования C#.

1. **Заключение**

Перед каждым из членов команды задача групповой работы впервые была поставлена при работе именно над этим проектом. Тем не менее, практически сразу после того, как мы узнали, что будем работать над общим проектом, решили, что наиболее подходящий язык для реализации проекта - C#, средство разработки – Microsoft Visual Studio. Именно это средство разработки подходило нам по характеру поставленной задачи. Так же немаловажен тот факт, что именно с этим средством разработки и языком программирования всем участникам проекта приходилось сталкиваться наибольшее число раз. Несмотря на то, что формат разработки подразумевает абсолютно чёткое распределение ролей, каждый участник, которому требовалась какая-либо помощь, получал её от остальных, что очень способствовало грамотной работе. Иногда возникали разногласия вследствие разного видения ситуации и задачи разными участниками проекта. Чаще всего подобные разногласия решались достижением какого-либо компромисса, в остальных случаях помогало взаимодействие с преподавателем, расставлявшее по местам и прояснявшее многие моменты по ходу разработки. Проект был реализован при помощи технологии итеративной разработки, что заметно упрощало работу над ним в целом, а использование такого сервиса контроля версий, как GitHub, позволяло осуществлять работу над проектом даже в те моменты, когда общий сбор и полноценное взаимодействие были невозможны по тем или иным причинам. За время нашей совместной деятельности было много различных хороших и плохих моментов, мы вздорили, мирились, спорили, но это очень сильно помогло научиться понимать друг друга, идти на компромиссы и работать в нужном русле. Каждый получил огромное количество необходимого в нашей дальнейшей профессиональной жизни опыта. В конечном счёте, все получили набор положительных впечатлений от работы в команде, а все тягости и невзгоды, которые легли на наши плечи, померкли перед фактом того, что мы смогли многому научиться за время нашей работы.