**Декомпозиция системы**

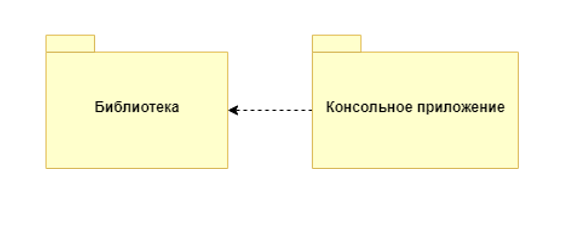
По ТЗ разрабатываемое ПО представлено двумя крупными подсистемами: библиотекой и консольным приложением для вызова функций библиотеки. Связь этих подсистем представлена на рисунке 1.

Рисунок 1. Подсистемы программы CCTIC

В соответствии с ТЗ консольное приложение должно:

* Считывание данных из файла;
* Проверка на недостающие элементы в входных данных;
* Сохранение результатов в файле.

Каждое из этих требований реализуют отдельные модули в составе приложения:

* FileParser — для чтения текстовых файлов;
* CompositionCheck — для проверки недостающих данных;
* FileSave — для сохранения результатов в файле.

Таким образом, работа приложения представлена работой четырёх модулей (рис. 2).

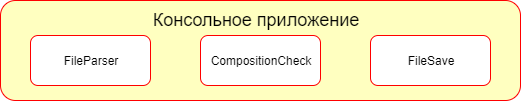


Рисунок 2. Модули консольного приложения

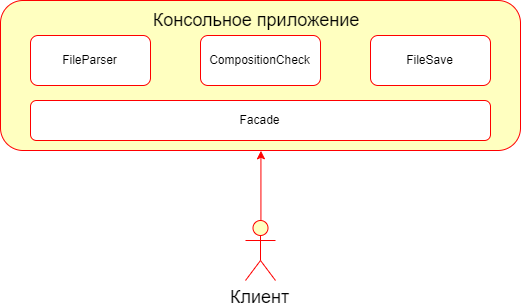
Работа клиента с модулями осуществляется с помощью паттерна Фасад (рис. 3).

Рисунок 3. Модель взаимодействия клиента с приложением

Интерфейс предоставляет клиенту публичный метод execute, принимающий в качестве аргумента путь к файлу.

Конфигурация приложения, т.е. установка необходимых модулей в составе приложения, осуществляется паттерном Строитель.

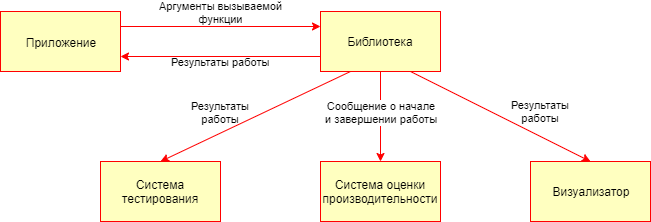
Контроль корректности работы системы выполняют система модульного тестирования, визуализатор и система проверки производительности, работающие отдельно от библиотеки и приложения. Связь между приложением и данными системами осуществляется посредством файлового обмена и обмена сообщениями. Общая схема работы всех составляющих изображена на рисунке 4.

Рисунок 4. Диаграмма структурной декомпозиции

**Сценарий запуска**

Рассмотрим работу системы на конкретном примере.

Сценарий 1:

*Шаг 1*. Ввод начальных данных. Пользователь вводит в консольное приложение путь к файлу с исходными данными.

*Шаг 2*. Считывание данных из файла. Приложение считывает данные и приводит их к нужному формату для дальнейшей проверки и работе с ними.

*Шаг 3*. Проверка недостающих элементов. Пусть модуль вернул значение больше нуля. Это значит, что в начальных данных есть недостающие элементы.

*Шаг 4.* Вызов аппроксимации. Вызов библиотечной функции аппроксимации для заполнения недостающих элементов.

*Шаг 5*. Сохранение в файл. Приложение выводит результат работы в файл.

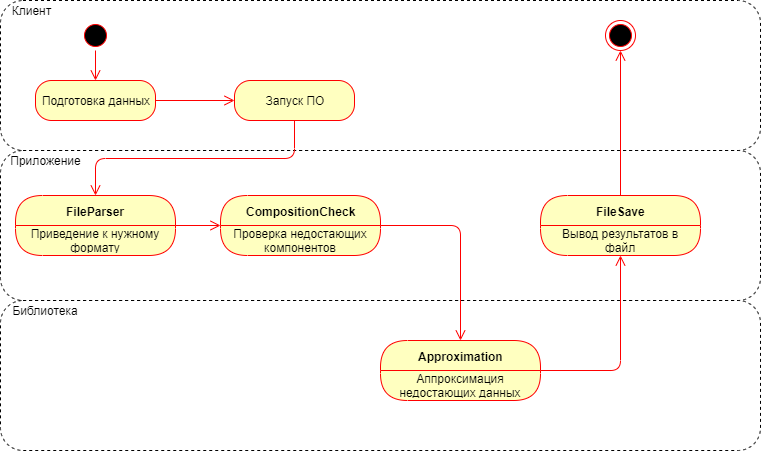
Диаграмма активностей, отражающая данный сценарий, изображена на рисунке 5.

Рисунок 5. Диаграмма активностей (Сценарий 1)

Сценарий 2:

*Шаг 1*. Ввод начальных данных. Пользователь вводит в консольное приложение путь к файлу с исходными данными.

*Шаг 2*. Считывание данных из файла. Приложение считывает данные и приводит их к нужному формату для дальнейшей проверки и работе с ними.

*Шаг 3*. Проверка недостающих элементов. Пусть модуль вернул значение равное нулю. Это значит, что в начальных данных нет недостающих элементов.

*Шаг 4.* Вызов предсказания. Вызов библиотечной функции предсказания для симуляции движения потока.

*Шаг 5*. Сохранение в файл. Приложение выводит результат работы в файл.

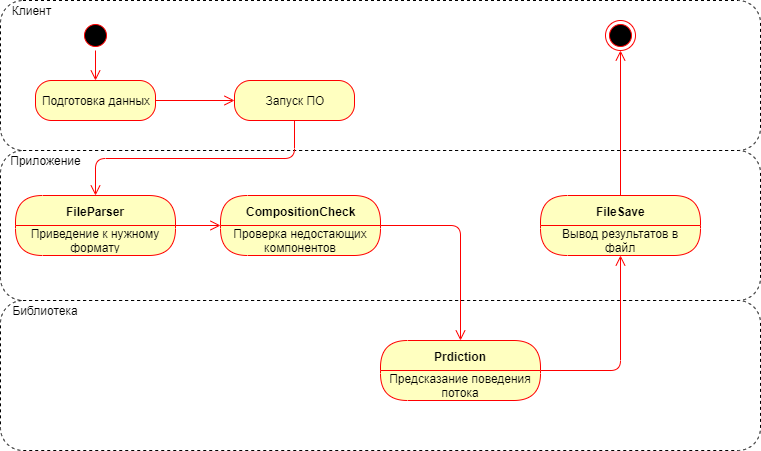
Диаграмма активностей, отражающая данный сценарий, изображена на рисунке 6.

Рисунок 6. Диаграмма активностей (Сценарий 2)

Сценарий 3:

*Шаг 1*. Ввод начальных данных. Данные вводятся автоматически на основе того, какие данные пришли вход и вышли на выход в сценарии 1 или 2.

*Шаг 2*. Считывание данных из файла. Приложение считывает данные и приводит их к нужному формату для дальнейшей работы с ними.

*Шаг 3*. Генерация изображения. По полученным данным генератор изображения создаёт визуальную интерпретацию скоростей потока.

*Шаг 4*. Сохранение в файл. Изображение сохраняется в файле.

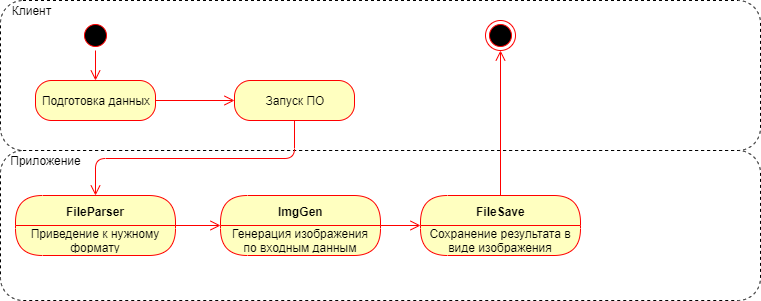
Диаграмма активностей, отражающая данный сценарий, изображена на рисунке 7. 

Рисунок 7. Диаграмма активностей. Сценарий 3

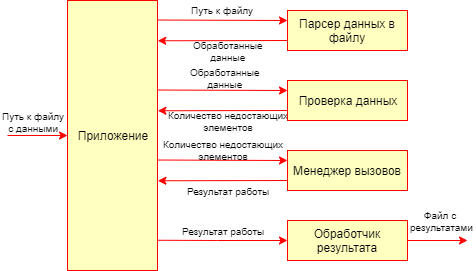


Рисунок 8. Диаграмма передачи данных