Развитие графоориентированного каркаса для обеспечения описания процессов проектирования технических объектов

Место проведения:
Продолжительность: *7 минут*

Тришин Илья Вадимович

МГТУ им. Н.Э. Баумана

Россия, Москва, – 24 марта 2022 г.

Содержание доклада

Введение

Постановка задачи

Архитектура программной реализации

Введение

↳ Пример процесса проектирования некоторого технического устройства

Дана задача – спроектировать технический объект. Как это сделать?
В качестве технического объекта рассмотрим радиоуправляемого робота с пилой на подвижном манипуляторе.

В его конструкции можно выделить следующие элементы:

- → бронированный корпус;
- → манипулятор;
- → пила;
- → колёса;
- → двигатели;
- → привод пилы;
- → привод колёс;
- → батарейный отсек;
- → радио-электронные компоненты;
- → …

Рис. 1: Пример проектируемого объекта

Введение

↳ Пример процесса проектирования некоторого технического устройства

- Перед началом проектирования формулируется техническое задание.
- В начале целесообразно проектировать элементы конструкции, не зависящие от других.

Рис. 2: Проектирование мобильного робота по компонентам

Введение

↳ Пример процесса проектирования некоторого технического устройства

- Затем проектируются элементы конструкции, конфигурация которых зависит от других.
- Параметрами проектирования подобных элементов являются характеристики других (геометрические, кинематические, мощностные и проч.)

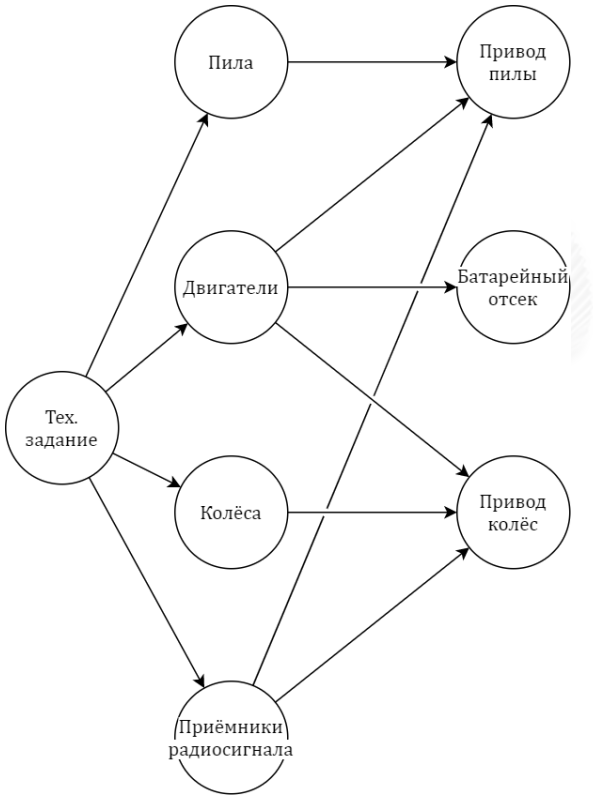


Рис. 2: Проектирование мобильного робота по компонентам

Введение

↳ Пример процесса проектирования некоторого технического устройства

- Затем проектируются подсистемы, объединяющие в себе различные элементы, сборочные узлы.
- Кроме того, проектируются элементы, которые должны содержать в себе другие сборочные узлы (корпуса и т.п.).

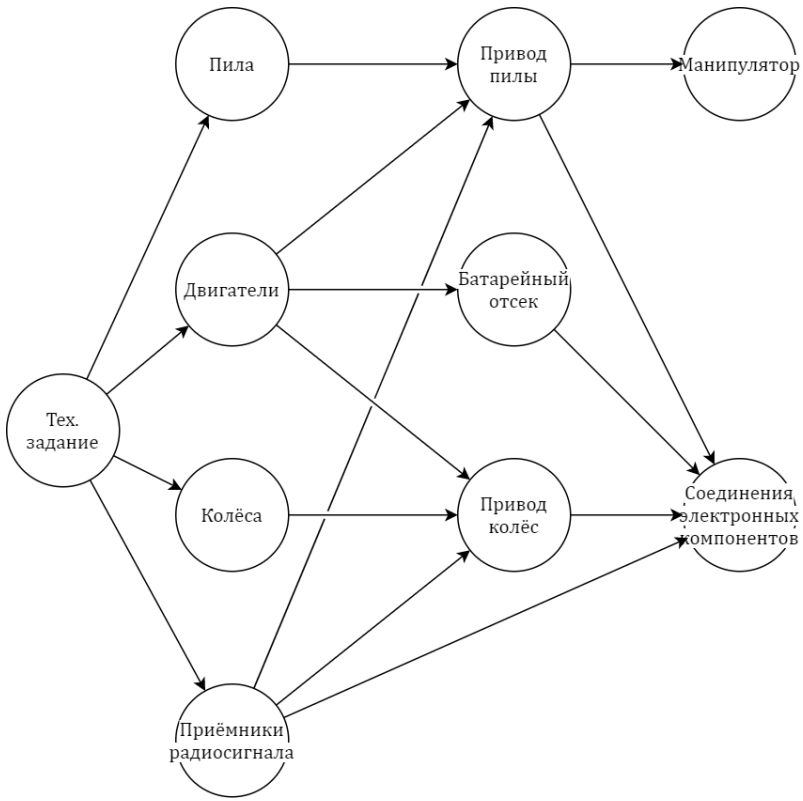


Рис. 2: Проектирование мобильного робота по компонентам

Введение

↳ Пример процесса проектирования некоторого технического устройства

- Затем проектируются подсистемы, объединяющие в себе различные элементы, сборочные узлы.
- Кроме того, проектируются элементы, которые должны содержать в себе другие сборочные узлы (корпуса и т.п.).

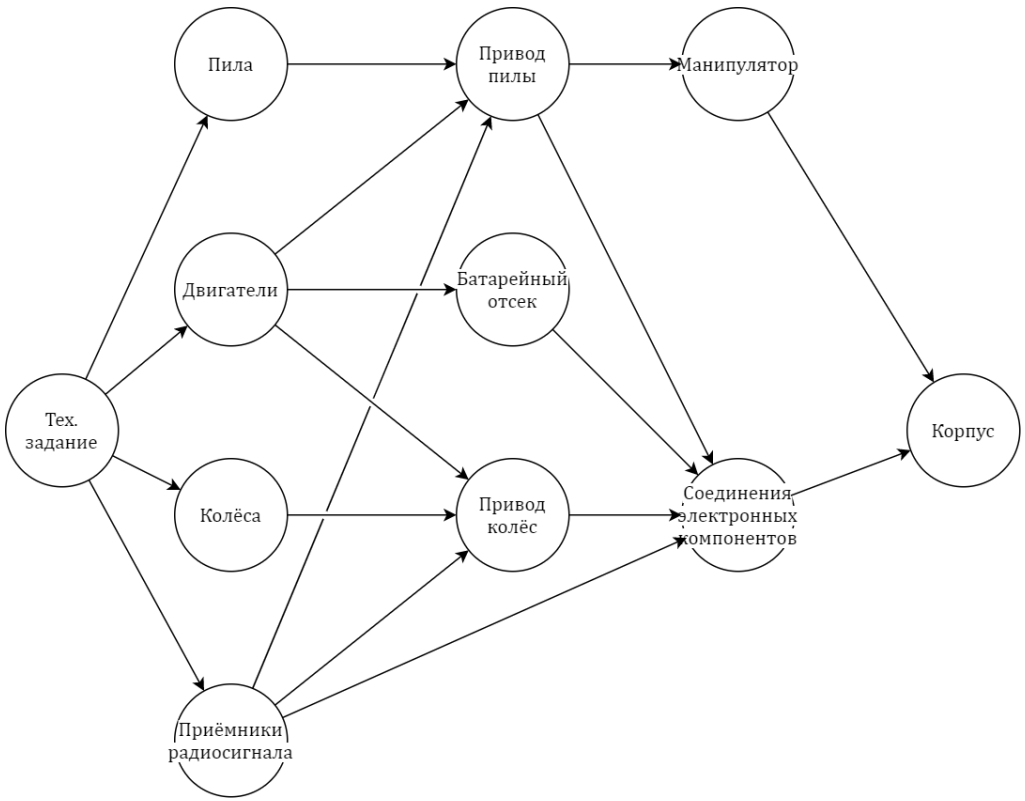


Рис. 2: Проектирование мобильного робота по компонентам

Утверждение 1

Процесс проектирования некоторого технического объекта удобно представлять в виде графа.

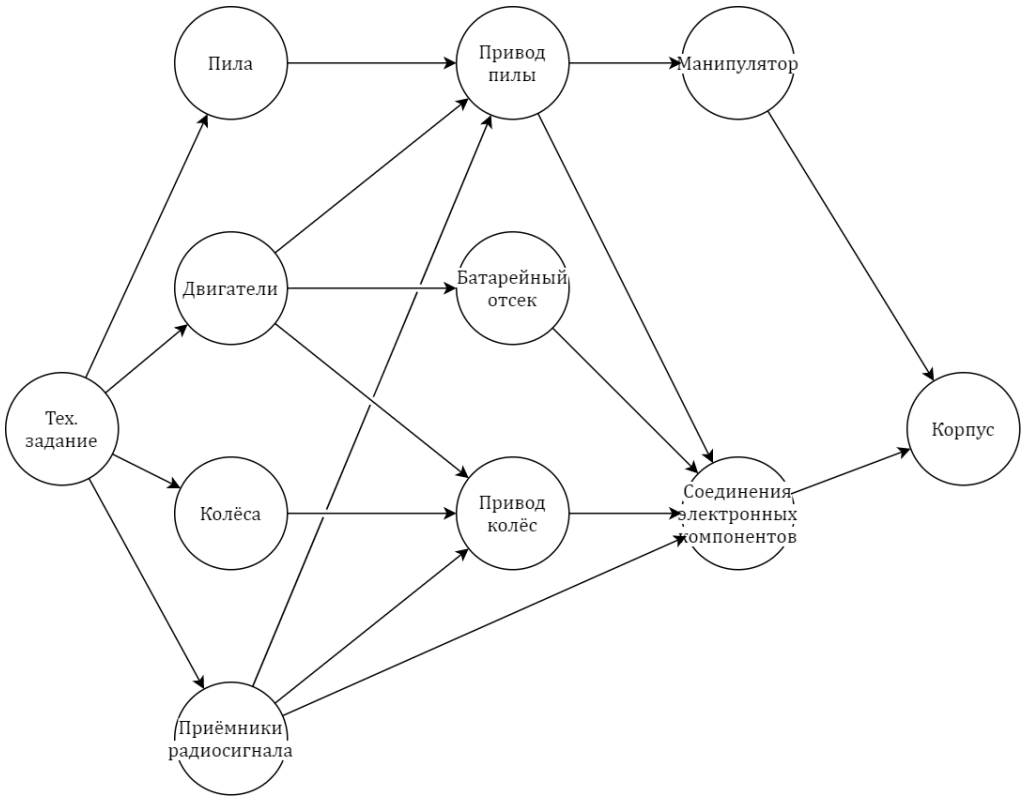


Рис. 2: Проектирование мобильного робота по компонентам

Графоориентированный подход – подход к описанию процесса решения сложной вычислительной задачи, при котором отдельные вычислительные процессы выстраиваются в ориентированный граф.

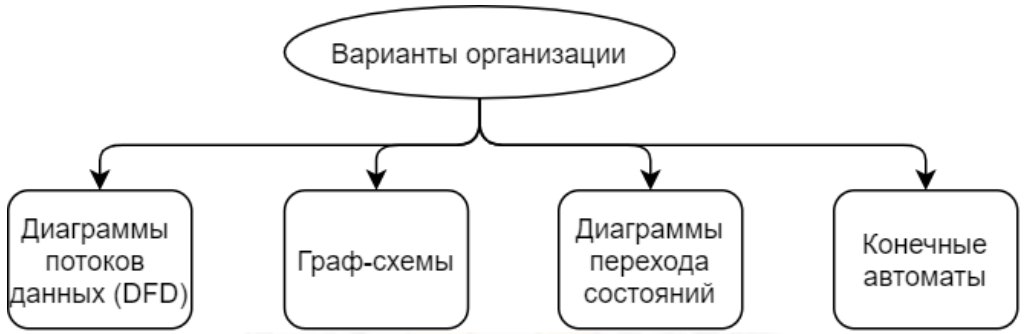


Рис. 3: Различные формы организации процессов проектирования в виде графа[1]

В данной работе внимание сосредоточено на подходе, реализованном в разработанном Соколовым А.П. и Першиным А.Ю. графоориентированном программном каркасе GBSE.

[1] Соколов А.П. Голубев В.О. Система автоматизированного проектирования композиционных материалов. Часть 3. Графоориентированная методология разработки средств взаимодействия пользователь–система // Известия СПбГЭТУ «ЛЭТИ». 2021. № 2.

В основе описываемого подхода следующие утверждения:

- Узлу ориентированного графа соответствует *состояние данных* – некоторый строго определённый набор именованных переменных фиксированного типа, характерных для решаемой задачи;
- Ребру ориентированного графа соответствует процесс преобразования одного состояния данных в другое.

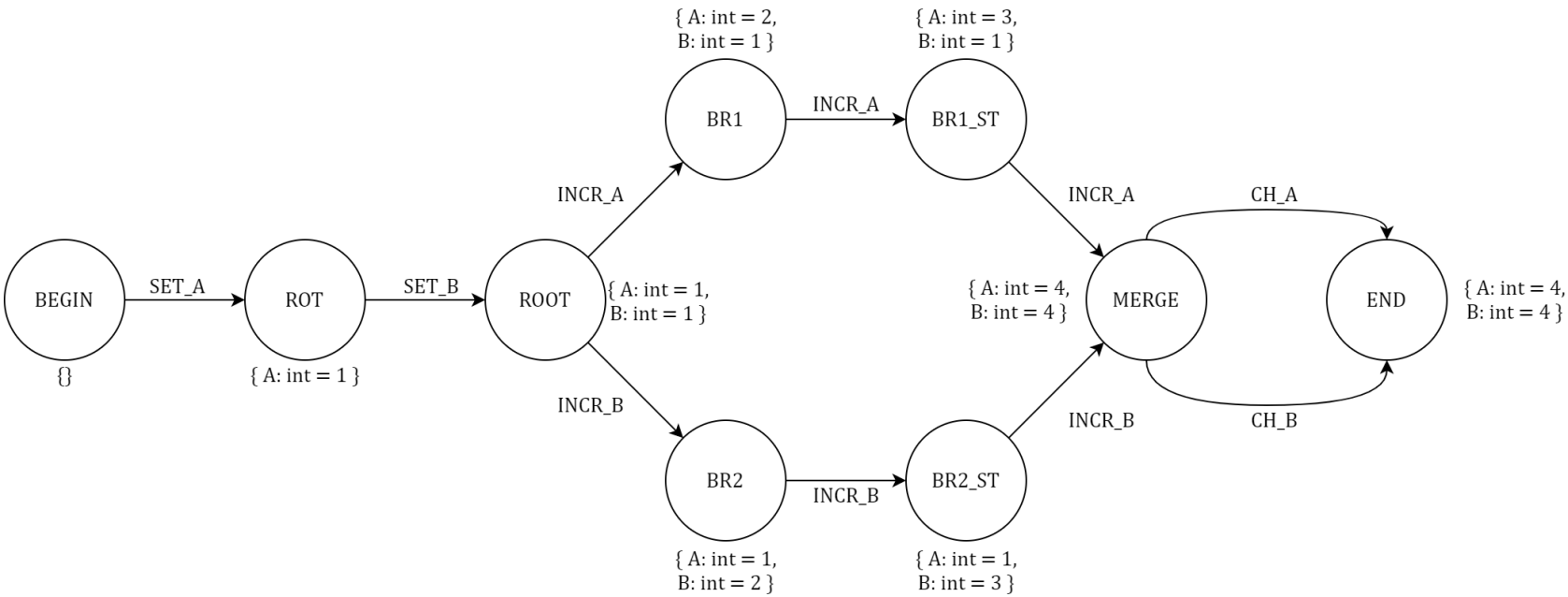


Рис. 4: Пример графовой модели вычислительного процесса

Обозначение 1

Функция-обработчик – функция, осуществляющая преобразование данных из одного состояния в другое.

Обозначение 2

Функция-предикат – функция, проверяющая корректность входного набора данных, соответствие их входному состоянию.

Обозначение 3

Функция-селектор – функция, отвечающая в процессе обхода графовой модели за выбор тех рёбер, которые необходимо выполнить на следующем шаге в соответствии с некоторым условием.

Замечание 2.1

Функции-селекторы реализуют условное выполнение ветвей графовой модели.

Замечание 2.2

Рёбра, относящиеся к разным ветвям графовой модели могут быть выполнены параллельно.

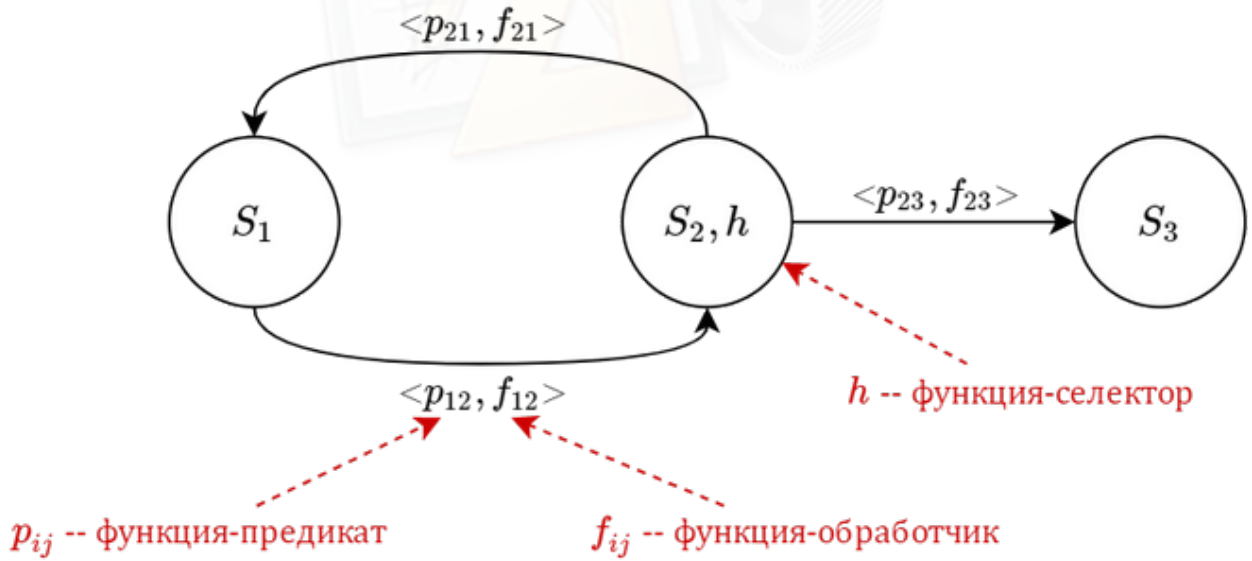


Рис. 5: Пример графовой модели с использованием селекторов

Объект исследований

процесс проектирования технического объекта

Цель исследования

предложить и реализовать новую структуру данных для хранения графовых моделей в рамках графоориентированного каркаса

Задачи исследования

- → Сравнить подходы к реализации графоориентированного подхода к решению задач проектирования на примере нескольких существующих программных комплексов
- → Исследовать программную структуру модуля каркаса GBSE, отвечающего за струкутуру графовых моделей
- → Определить требования к структуре данного модуля
- → Разработать новую структуру, которая бы отвечала сформулированным требованиям

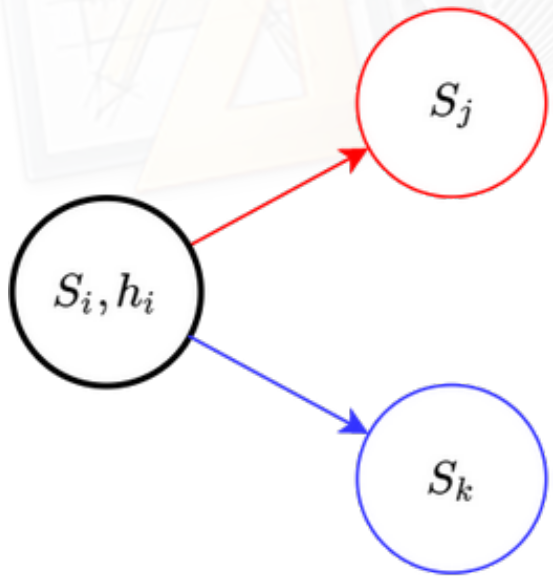
- Должна обеспечиваться поддержка актуальной версии формата aDOT [2].
- Реализуемая структура данных должна содержать в себе все узлы и рёбра, относящиеся к данной графовой модели.
- Топология графовой модели должна описываться матрицей смежности.
- Реализуемая структура данных должна предоставлять интерфейс для доступа к данным о топологии.

[2] Соколов А.П. Описание формата данных aDOT (advanced DOT) [Электронный ресурс]. Облачный сервис SA2 Systems. [Офиц. сайт]. 2020. (дата обращения 05.03.2020) URL: https://sa2systems.ru/nextcloud/index.php/f/403526.

Постановка задачи

↳ Требования к программной реализации структуры данных узла графовой модели

- → Реализуемая струкутра данных должна содержать в себе:
  - описание состояния данных (в некотором внутреннем формате);
  - стратегию параллельного исполнения[3];
  - функцию-селектор (или ссылку на неё).
- → Реализуемая структура данных должна предоставлять интерфейс для:
  - назначения стратегию параллельного выполнения;
  - назначения функции-селектора;
  - вызова функции селектора с заданным набором данных.

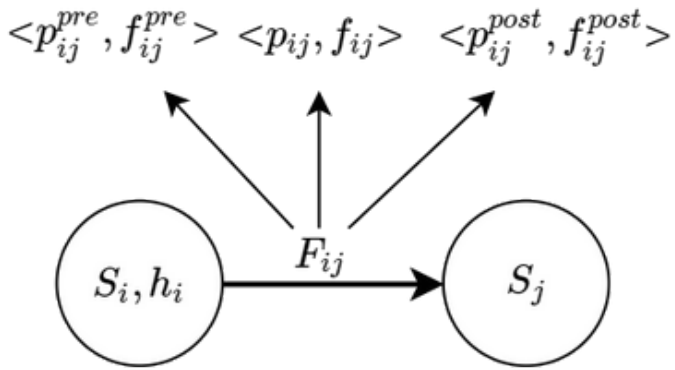


[3] т.е. данные о том, как параллельно исполнять исходящие из данного узла рёбра

Постановка задачи

↳ Требования к программной реализации структуры данных ребра графовой модели

- → Реализуемая структура должна содержать в себе до трёх структур данных (или ссылок на них) типа «морфизм» – препроцессор, обработчик и постпроцессор.
- → Функциональная структура данных типа «морфизм» должна содержать в себе:
  - ссылку на функцию-предикат;
  - ссылку на функцию-обработчик;
- → Реализуемая структура должна предоставлять интефейс для:
  - назначения препроцессора, обработчика и постпроцессора;
  - вызова хранящихся в ней «морфизмов» с заданным входным набором данных.



В библиотеке comsdk, являющейся реализацией описанного выше подхода на языке C++, помимо прочих описаны следующие структуры данных:

- Anymap – ассоциативный массив, позволяющий хранить в себе разнотипные данные;
- ActionItem – функциональный объект, реализующий функцию-обработчик;
- ActionItemContext – объект, осуществляющий запуск функций-обработчиков и хранящий данные об их выполнении;
- Predicate – функциональный объект, являющийся обёрткой над некоторой функцией, ставящей в соответствие входному набору данных логическое значение (0 или 1);

Структуры данных разрабатывались с учётом возможностей стандарта C++-11 и библиотеки Standart Template Library (STL). Для описания разработанных структур данных был использован язык графического описания UML (Unified Modeling Language).

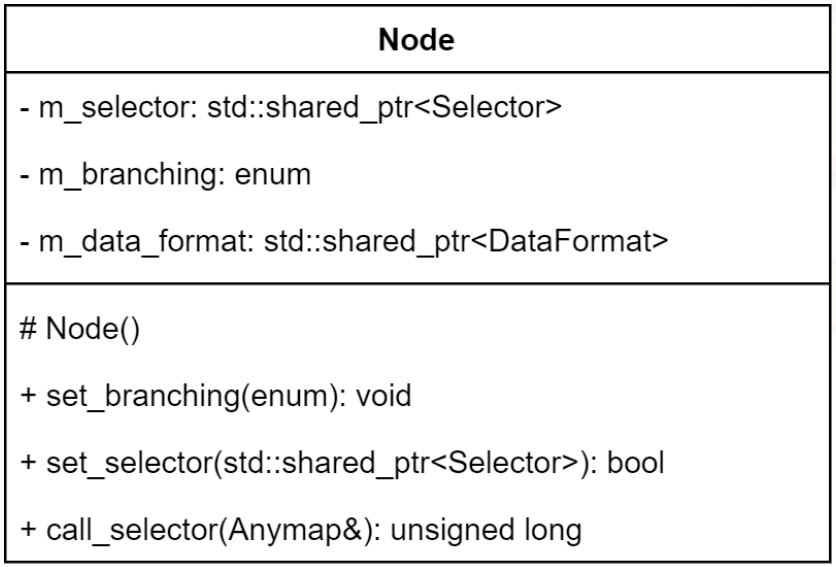


Рис. 6: UML-диаграмма класса узла графа

- Использует умные указатели (shared_ptr) – эффективное использование памяти.
- Защищённый конструктор – объекты данного класса создаются только в пределах класса графовой модели.
- Интерфейс и информационные поля соответствуют требованиям.

Архитектура программной реализации

↳ Предлагаемая структура данных ребра графовой модели

- Использует умные указатели (shared_ptr) – эффективное использование памяти.
- Защищённый конструктор – объекты данного класса создаются только в пределах класса графовой модели.
- Интерфейс и информационные поля соответствуют требованиям.
- Для прохода по ребру необходимо назначить ему хотя бы одну пару предикат-обработчик (объединённых в структуру данных Morphism) с помощью метода set_function.

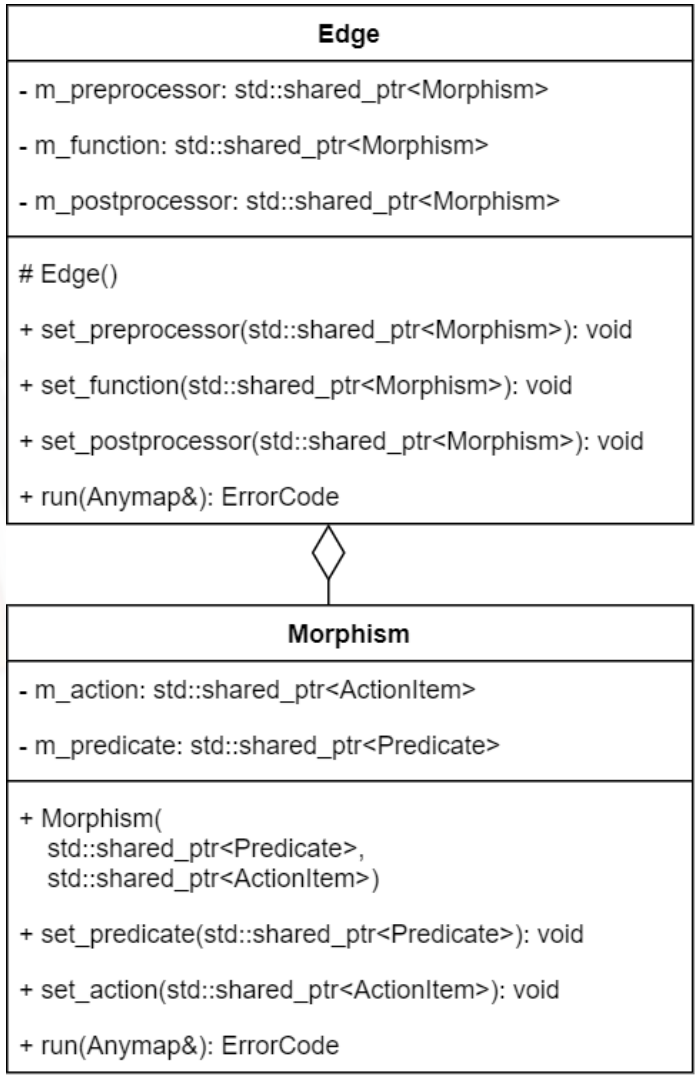


Рис. 7: UML-диаграмма класса ребра графа

Архитектура программной реализации

↳ Предлагаемая структура данных графовой модели

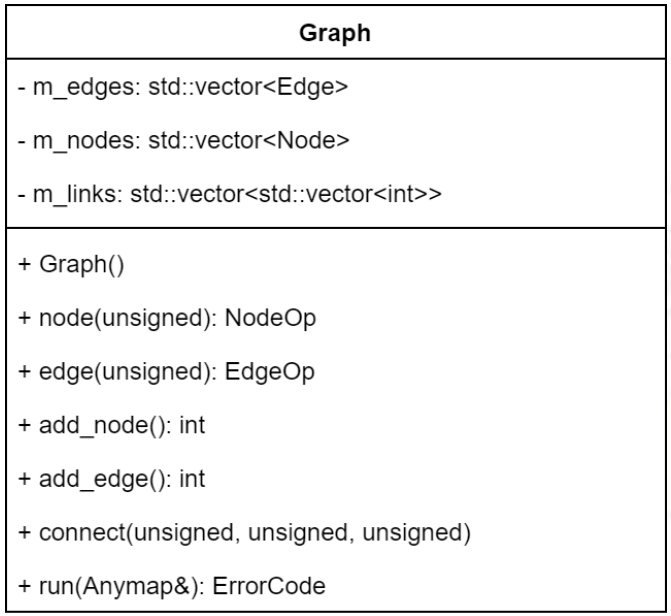


Рис. 8: UML-диаграмма класса графа

- Хранит в себе узлы и рёбра графовой модели.
- Хранит в себе динамическую матрицу смежности.
- Позволяет добавлять узлы и рёбра и связывать их.
- Позволяет запустить обход графовой модели с заданным набором входных данных.

- NodeOp – сокращение от “Node Operation”.
- Дополнительная структура данных для операций с узлами.
- Хранит только индекс узла – эффективное использование памяти.
- Позволяет получить соседние узлы для данного узла;
- Позволяет получить входящие и исходящие рёбра для данного узла;
- Позволяет при необходимости обратиться к интерфейсу самого узла.

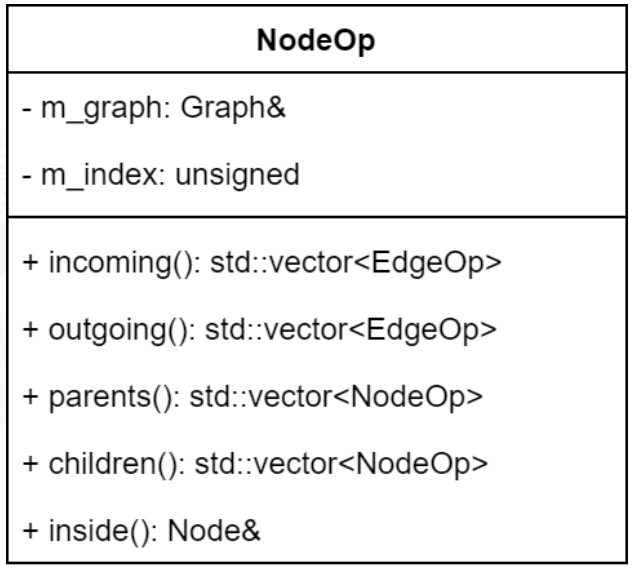


Рис. 9: UML-диаграмма класса операции с узлом

- EdgeOp – сокращение от “Edge Operation”
- Дополнительная структура данных для операций с рёбрами.
- Хранит только индекс ребра – эффективное использование памяти.
- Позволяет получить начальный и конечный узлы для данного ребра.
- Позволяет при необходимости обратиться к интерфейсу самого ребра.

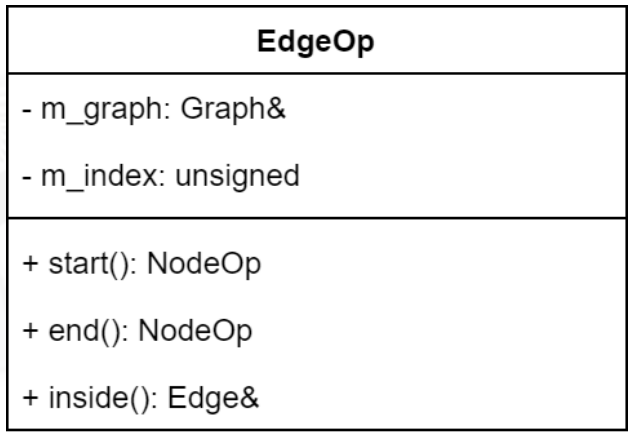


Рис. 10: UML-диаграмма класса операции с ребром

Заключение

- Изучены разные подходы к организации вычислительных процессов в программных комплексах;
- Изучена программная структура комплекса GBSE;
- Разработана новая структура графового модуля GBSE:
  - Разработанная структура соответствует новому планируемому формату;
  - Разработанная структура позволяет устранить недостатки старой версии;

Спасибо за внимание!

Вопросы?

Приложение. Основная терминология

ПО Программное обеспечение. 6