Список используемых сокращений и терминов

«График матчей» – разрабатываемая автоматизированная система.

Лог – текстовый файл, содержащий протокол работы АС.

Метрика – вектор, характеризующий качество решения.

Неполные данные – данные, которые нельзя считать достаточными для работы системы.

Некорректные данные – данные, формат которых не соответствует установленным требованиям.

Описание проблемы

В настоящий момент времени расписание матчей чемпионата Нижегородской области по мини-футболу составляется ответственным лицом вручную. Обработка ответственным большого количества информации, необходимой для формирования календаря, влечёт значительные временные затраты, высокую вероятность нестыковок, что в свою очередь, приводит к итеративности процесса. Автоматизированная система «График матчей» разрабатывается для сокращения времени, затрачиваемого на составление графика.

Функциональные требования:

1. Система должна получать исходные данные, достаточные для решения задачи.
2. Система должна осуществлять проверку исходных данных на предмет:
   1. соответствия формата (требования к формату исходных данных предъявляются в процесс НИР);
   2. соответствия требованиям задачи (полнота и непротиворечивость данных).
3. Система должна получать на вход корректную полную информацию и строить график спортивных мероприятий чемпионата (выходная информация).
4. Система должна генерировать протокол работы в виде текстового файла, где указываются:
   1. результат проверки исходных данных;
   2. результат работы алгоритма;
   3. в случае получения положительного результата – численные значения метрик.

Нефункциональные требования

1. Система предполагает расчёт календаря матчей не более чем для 100 команд единовременно.
2. В один временной слот возможно проведение только одной игры, что эквивалентно формированию календаря игр для одного стадиона.
3. Время проведения расчёта для 10 команд не должно превышать 10 минут, для 20 команд – 30 минут и т.д.
4. Система реализует не более четырёх различных алгоритмов построения графика матчей.

Общие решения

Диаграмма работы АС «График матчей» представлена на .



Рисунок 1. Схема работы АС «График матчей»

Блок Парсер преобразует исходные данные в программную модель, которая будет использована системой для проведения расчёта.

Блок Анализатор выполняет проверку полученной программной модели на достаточность данных для проведения расчёта.

Блок Solver выбирает алгоритм(ы) решения, производит выбор лучшего результата и реализует вывод расписания и протокола работы.

Лог представляет собой текстовый документ, в который выводятся:

1. Сведения об успешности выполнения расчёта;
2. Численные значения метрики;
3. В случае, когда решение не построено, данные о причине, вызвавшей сбой, (некорректность, несовместность, неполнота данных, внутренняя ошибка).

Исходные данные поступают в систему в формате .xml и представляют собой файл исходных данных задачи и файл конфигурации. Структура xml-документа исходных данных задачи представлена на :



Рисунок 2. Структура xml-документа с исходными данными задачи

Пример исходных данных:

<shedule>

<championship start="22.11.2019" end="22.06.2020">

<day>пн</day>

<day>пт</day>

<coupleGames>2</coupleGames>

<decree>05.04.2019</decree>

</championship>

<stadium name="ФОК Приокский">

<minGame>1</minGame>

<maxGame>4</maxGame>

<time>19:00</time>

<time>20:00</time>

<time>21:00</time>

<time>22:00</time>

</stadium>

<command name ="Родина (Дзержинск)">

<wishes>

<time>21:00</time>

<time>21:00</time>

</wishes>

</command>

<command name ="Главспорт">

<wishes />

</command>

<command name ="Радий">

<wishes>

<day>пн</day>

<time>21:00</time>

</wishes>

</command>

.

.

.

</shedule>

Конфигурационный файл содержит настройки преобразования исходных данных задачи в программную модель, настройки проверки программной модели, настройки блока Solver (выбор алгоритма решения текущей задачи, выбор метрики для расчета). Структура xml-документа файла конфигурации представлена на :



Рисунок 3. Структура xml-документа конфигурационного файла

Выходными данными (решением) является турнирная таблица, представленная на html-странице.

В качестве метрики принимается вектор вещественных чисел. Детали, связанные с функциями расчёта метрики будут уточнены в пояснительной записке.

Требования к тестовой инфраструктуре

Тестовая инфраструктура необходима для проверки корректности работы алгоритмов системы, их настройки и сравнения между собой.

Исходными данными для тестовой инфраструктуры являются:

1. Пул задач – директория, в которой находится набор тестовых примеров (xml-документов с исходными данными);
2. Алгоритм(ы) из блока Solver;
3. Метрики, которые необходимо рассчитать.

Выходные данные представляют собой сводную таблицу по результатам работы системы на наборе тестовых примеров, которая позволяет оценить эффективность алгоритмов блока Solver на разных задачах.

Сценарии работы

Actors:

1. Пользователь – ответственный за формирование календаря матчей;
2. Администратор – настройщик системы, создаёт/указывает конфигурационный файл;
3. Research-исследователь – разработчик, использующий тестовую инфраструктуру.

Сценарий 1 – данные верны:

1. Администратор создаёт/указывает конфигурационный xml-файл.
2. Пользователь вводит/указывает xml-файл с исходными данными.
3. Система принимает на вход xml-файлы конфигурации и исходных данных.
4. Блок Парсер преобразует полученный xml-файл с исходными данными в программную модель.
5. Блок Парсер передаёт программную модель блоку Анализатор.
6. Блок Анализатор считывает из конфигурационного файла параметры проверки программной модели.
7. Блок Анализатор проверяет корректность полученной программной модели.
8. Блок Анализатор передаёт проверенную модель блоку Solver.
9. Блок Solver считывает из конфигурационного файла информацию о выбранных алгоритмах и метрике.
10. Блок Solver производит поиск решения разными алгоритмами и вычисление метрики.
11. Блок Solver производит сравнение вычисленных метрик и выбирает лучшее решение.
12. Блок Solver осуществляет вывод полученного решения на html-страницу, вывод протокола работы и численные значения метрики в лог-файл.
13. Пользователь получает решение на html-странице и текстовый лог-файл.

Сценарий 2 – данные некорректны:

1. Администратор создаёт/указывает конфигурационный xml-файл.
2. Пользователь вводит/указывает xml-файл с исходными данными.
3. Система принимает на вход xml-файлы конфигурации и исходных данных.
4. Блок Парсер пытается преобразовать полученный xml-файл с исходными данными в программную модель.
5. Блок Парсер обнаруживает ошибку в файле исходных данных.
6. Блок Парсер помещает в лог-файл информацию об остановке работы системы и указывает локализацию некорректных данных.
7. Пользователь получает текстовый лог-файл.

Сценарий 3 – данные неполны:

1. Администратор создаёт/указывает конфигурационный xml-файл.
2. Пользователь вводит/указывает xml-файл с исходными данными.
3. Система принимает на вход xml-файлы конфигурации и исходных данных.
4. Блок Парсер преобразует полученный xml-файл с исходными данными в программную модель.
5. Блок Парсер передаёт программную модель блоку Анализатор.
6. Блок Анализатор считывает из конфигурационного файла параметры проверки программной модели.
7. Блок Анализатор проверяет корректность полученной программной модели.
8. Блок Анализатор обнаруживает в программной модели несоответствие параметрам из конфигурационного файла.
9. Блок Анализатор помещает в лог-файл информацию об остановке работы системы и указывает локализацию неполных данных.
10. Пользователь получает текстовый лог-файл.

Сценарий 4 – данные несовместны:

1. Администратор создаёт/указывает конфигурационный xml-файл.
2. Пользователь вводит/указывает xml-файл с исходными данными.
3. Система принимает на вход xml-файлы конфигурации и исходных данных.
4. Блок Парсер преобразует полученный xml-файл с исходными данными в программную модель.
5. Блок Парсер передаёт программную модель блоку Анализатор.
6. Блок Анализатор считывает из конфигурационного файла параметры проверки программной модели.
7. Блок Анализатор проверяет корректность полученной программной модели.
8. Блок Анализатор передаёт проверенную модель блоку Solver.
9. Блок Solver считывает из конфигурационного файла информацию о выбранных алгоритмах и метрике.
10. Блок Solver производит поиск решения разными алгоритмами и вычисление метрики.
11. Блок Solver обнаруживает несовместность исходной системы ограничений задачи.
12. Блок Solver помещает в лог-файл информацию об остановке работы системы и указывает несовместные данные.
13. Пользователь получает текстовый лог-файл.

Сценарий 5 – тестовая инфраструктура:

1. Research-исследователь регистрирует алгоритм.
2. Research-исследователь создаёт конфигурационный файл формата .xml.
3. Research-исследователь указывает директорию, в которой находится набор тестовых данных.
4. Система принимает на вход xml-файлы конфигурации и тестовых данных.
5. Блок Парсер пытается преобразовать полученные xml-файлы с тестовыми данными в программные модели.
6. При нахождении ошибки в xml-файле тестовых данных блок Парсер помещает в лог-файл информацию о некорректности текущего тестового примера.
7. Блок Парсер передаёт набор корректных программных моделей блоку Анализатор.
8. Блок Анализатор считывает из конфигурационного файла параметры проверки программных моделей.
9. Блок Анализатор проверяет полноту полученных программных моделей.
10. При нахождении несоответствия параметрам из конфигурационного файла блок Анализатор помещает в лог-файл информацию о неполноте текущих тестовых данных.
11. Блок Анализатор передаёт набор проверенных моделей блоку Solver.
12. Блок Solver считывает из конфигурационного файла информацию о выбранных алгоритмах и метрике.
13. Блок Solver производит поиск решения разными алгоритмами и вычисление метрики.
14. Блок Solver производит запись вычисленных метрик в лог-файл.
15. Research-исследователь получает текстовый лог-файл, содержащий сводную таблицу по результатам работы системы на наборе тестовых примеров.