08148 17/18 Software Engineering ACW Report

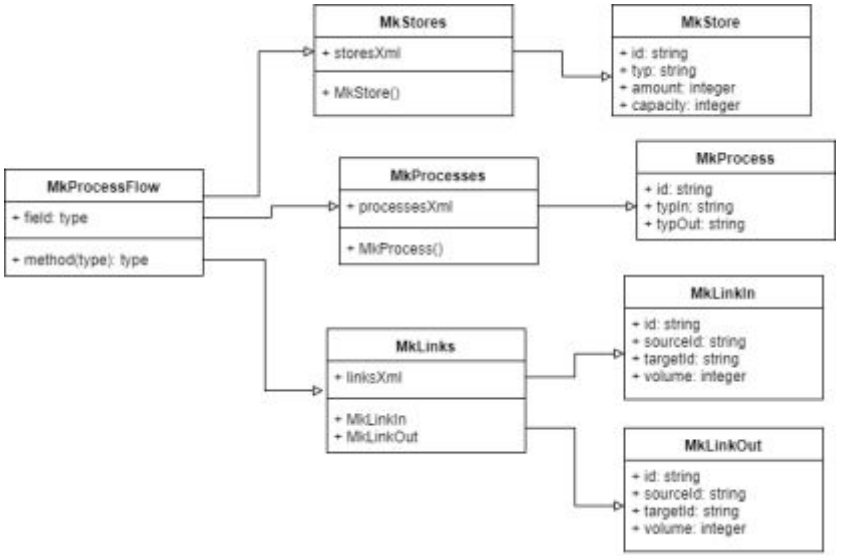
John Allison

Overview

The program's purpose is to mimic a 'process flow' diagram, that takes an input XML in this case 'flow.xml', then the program verifies that the formatting conforms correctly then reads the data into four tasks.
The first, flow creates a process flow structure which has various stores for input and output resource stores that can be limited, processes and links which links the resources to one or more process.
The remaining three are: load which loads the given, execute which executes the given integer and query which checks if the given query is in the current flow store.

Program Architecture & Design

The program's overall design and architecture is relatively simple due to the use of the below object oriented design patterns. The input file is an XML document which loads the inputs into a list where the `ProcessFlowFactory` parses the inputted data into XML nodes, below is a class diagram of this process.



The parsed objects are stored into a list 'inputs', then they are executed if successful in execution the result is then stored into a list 'results'. When all inputs are parsed, the results are then written into an XML document 'out.xml' through an XML serializer.

Object Oriented Design Patterns

This program uses many Object Oriented Design Patterns which is a type of Software Design Pattern that are used universally to make program creation simpler and reduces the need to 're-invent the wheel' when developing a program. This helps to achieve high cohesion and low coupling therefore making the program easier to iterate and maintain.
Some of the design patterns used are as follows:

- **Factory Method -** A factory method is a way to streamline the code implementation and streamlines object creation. This has been implemented in this program as `ProcessFlowFactory.cs` which handles XML input, validation, data parsing and finally output.

- **Template Methods -** Template Methods are methods that create a barebone version of an algorithm or class and allows its subclasses to override and iterate on parts without affecting the overall structure of the class. The Factory Method uses template methods to create the objects used in the factory.
  An implemented example in the program is:
  ```
  ProcessFlow processFlow = MkProcessFlow(inputXml);
  ```

  This uses calls from a separate class that creates the objects that ProcessFlow uses to execute. Each objects also contain subclasses that contain many Stores, Processes and Links.
  ```
  return new ProcessFlow(stores, processes, links);
  ```

- **Facade -** Facades are used to simplify the lower level systems but does not remove functionality. In the program's implementation, ProcessFlow objects use Stores, Processes and Links, but each of these have their own single class. For example `Link` have `LinkIn` which by design connect the input store to the process.
  Therefore this hides singular `LinkIn` behind a larger `Link`.