Design of VDisp Software

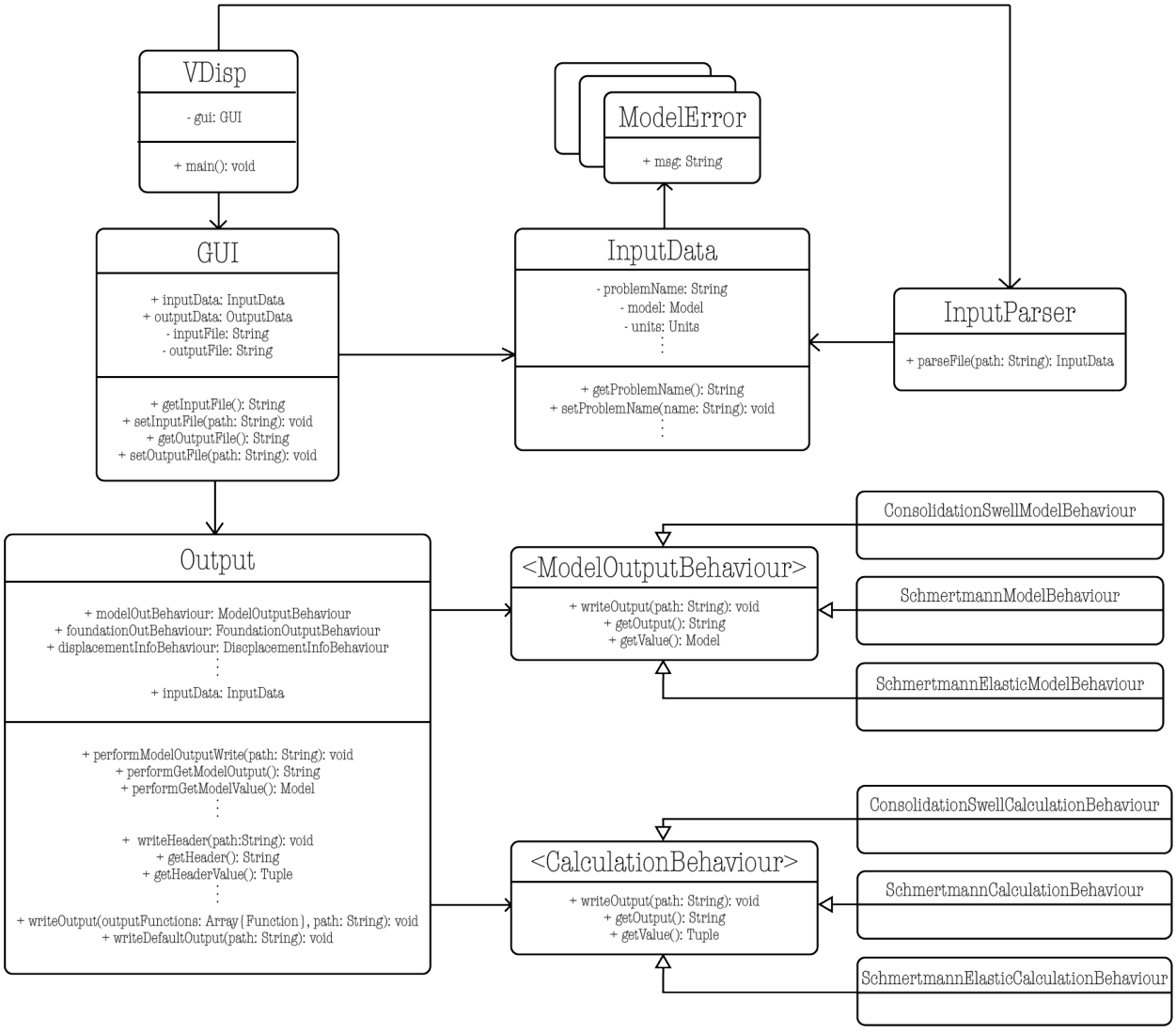
Emil Soleymani, Dr. Spencer Smith
McMaster University

August 19, 2022

The **VDisp** software was carefully designed to adhere to the *SOLID* software design principles. This document aims to outline the initial proposed design (which will be called the ideal design), the actual implemented design, and the limitations of the Julia programming language that lead to this drastic change.

Ideal Design

This section aims to describe the *ideal* design which was drafted for the **VDisp** software.



The diagram above has been condensed. *InputData* uses many Custom Exception classes which help identify specific problems in the input file format allowing for helpful and descriptive messages to be given to

the user. Furthermore, the **FoundationOutputBehaviour**, **DisplacementInfoBehaviour**, **ForcePointBehaviour** and **EquilibriumInfoBehaviour** interfaces (and the classes that implement them) have been left out of the diagram.

Examining the **ModelOutputBehaviour** interface is sufficient to understand the implementation of the excluded interfaces, while the **CalculationOutputBehaviour** is more complex than the other interfaces. The classes that implement **CalculationOutputBehaviour** are responsible for making method specific calculations, and return a set of values that is dependant on the method itself. This is why the **CalculationOutputBehaviour** *getValue()* function is said to return a **Tuple**. This **Tuple** contains different info based on the specified model, and classes that access this **Tuple** must be prepared to parse it.

The design pattern of the *Output* class and the interfaces it uses (all the interfaces that end in "*Behaviour*") is inspired by the *Duck example* in the opening chapter of this textbook on design patterns.