**P445 - Assignment P-06**

**Armadillo Island**

**Kevin Florian**

**Joshua Rowe**

**Erica Wright**

1. The heavy use of XML in our project greatly relieves our team from the traditional use of definite data structures in preforming routine tasks in our application. Since XML is our storage medium (in place of a database and plain old objects), we can take advantage of a nice .NET class, called XMLSerializer. In a sense, the XML serializer becomes the intermediate data structure as it parses XML and writes to an XML file. We will read input text from users and immediately write it to an XML format and deal with the XML from there on. Thus, we might not require an intermediate storage in some structure. We are not ruling out using definite data structures, but we may not even require them. In the case of relationships and attributes of certain classes, once again, if we rely on .NET prebuilt classes, we may not have to write our own, and, thus, further decreasing our need for plain data structures in our implementation.
2. Information that the program uses comes from two sources. The primary source is the user entering text information or selecting from pre-filled drop-down boxes (or other equivalent selection tool). The pre-filled selections will be generated from what the user has previously entered. The other option is input from a file in NEXUS format. From the original data that is input into the program, the data flows into XML format, which can then be parsed and then used to write to the NEXUS format. From the flow that was just mentioned, user input will be mapped to a text boxes with logic attached or a function that reads NEXUS and auto-fills data that the user could modify. Once information is entered to the user’s satisfaction, a selection to translate to NEXUS is chosen. Afterwards, information flows from XML to the NEXUS file. This would map to a function which would take care of such a translation and writing, or a subsystem of functions to take care of the work.
3. There are no external interfaces for our program to deal with in the form of databases or outward facing APIs. We do have a mock-up of the user interface.
   1. SEE MOCK-UP.
4. The main drivers of the program are the UI, subsystems to parse and translate information entered by the user or auto-filled by the program itself, and a subsystem to write translated information into the NEXUS format in which other phylogenetic programs can take advantage of.
5. \*Detail of individual breakdown\*
   1. Kevin on UI??
6. \*Our information\*