Milestone 1 (REDO)

Git Lucky:
Jomar Dimaculangan, Devon Honig, Mathew Merrick

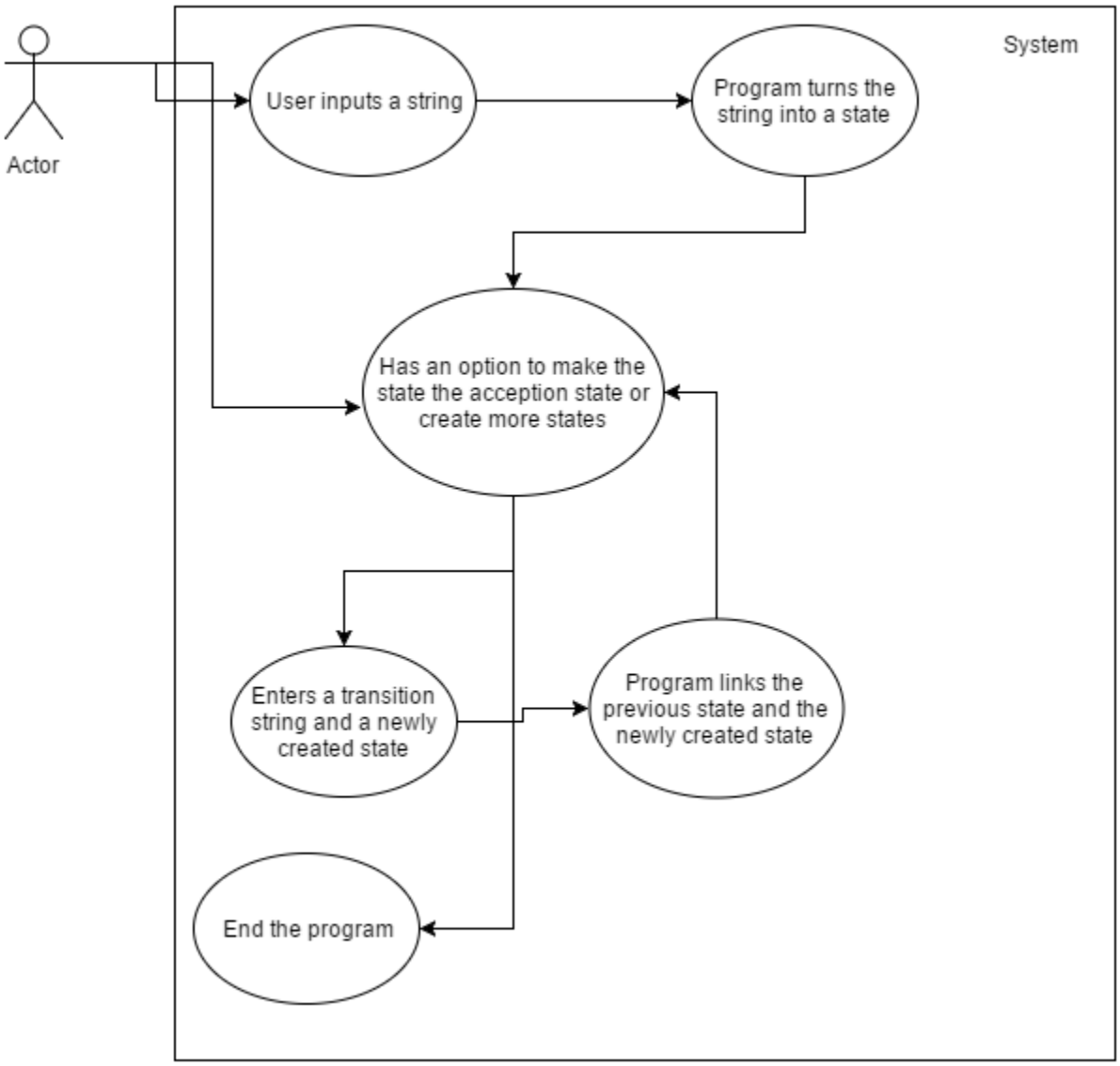
Finite State Machine Use Cases

1) Creation of the automata:
    a. Primary User:
        i. Student (programmer)
    b. Goal in Context:
        i. Build an automata according to the user's inputs from the keyboard and not an external file
        ii. The automata must include the five elements which are the states set, strings, transition sets, initial set, and accepting state set.
    c. Preconditions:
        i. A command line interface for the user to input the strings
        ii. The first state will be the initial state
        iii. The user can choose what the automata's accepting case would be
        iv. The automata must be deterministic
    d. Trigger:
        i. Once the program is running, a command line interface will be shown. From that, the user will input the initial string and the initial state will be made. After the user presses enter in their keyboard and the automata is done creating, other features are shown.
    e. Scenario:
        i. Student – Opens up the program
        ii. Student – Has an option to create an initial state and creates one
        iii. Program – The program will turn the string into a state and waits for the student to input strings for transitioning through the automata or waits for the student to create the accepting state
        iv. Student – Enters a string and creates another state which the student chose to be the accepting state
        v. Program – links the previous state with the newly created state using the string input.
        vi. Student ends the program

Milestone 1 (REDO)

Git Lucky:
Jomar Dimaculangan, Devon Honig, Mathew Merrick

Use Case 1 Diagram:

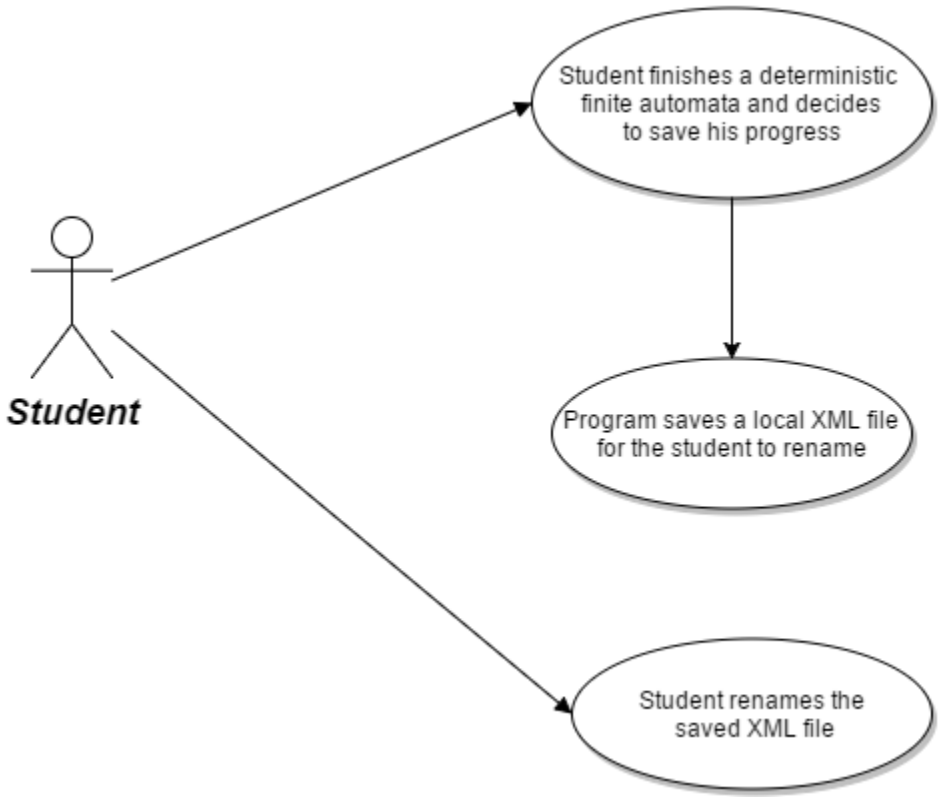


2) Save feature into a file
    a. Primary User:
        i. Student
    b. Goal in Context:
        i. The student should be able to save the automata in an XML file so the user doesn't have to start their automata from scratch every time
    c. Precondition:
        i. A command line function to save the current work to an XML file
        ii. Knowing where to save the file in the computer
    d. Trigger:

i. Once the user edits the program in any way (such as creating a transition string from two states), the user can call the save function and it edits their XML file

e. Scenario:
   i. Student – Finishes a deterministic automata using the function the first use case described.
   ii. Student – Decides to save the current progress in an XML file using the save function.
   iii. Program – Once the user saves the automaton, the program will create a local XML file and student is asked to rename it.

Use Case 2 Diagram:

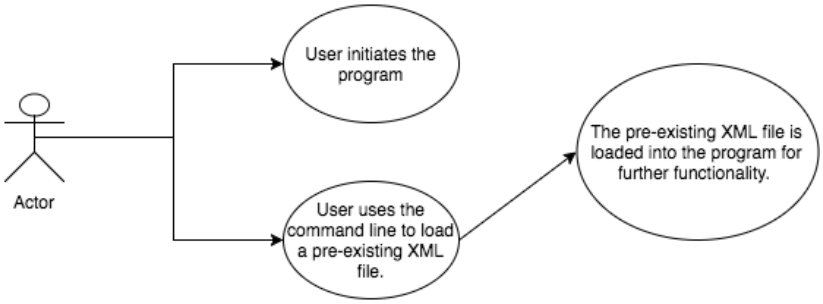


3) Create a Load Feature
   a. Primary User
      i. Student (Programmer)
   b. Goal in Context:
      i. The student(programmer) will be able to load an XML file as an extension of the program.
   c. Precondition:
      i. A command line function to load a needed XML file.
      ii. Knowing the location of the needed XML file.
      iii. Program logic for the automata, as a whole, and the separate load function must be in place.
   d. Trigger:

i. After the program is initiated, a command line function will be available to the user. The user will then be able to initiate the load function in order to load the needed XML file.

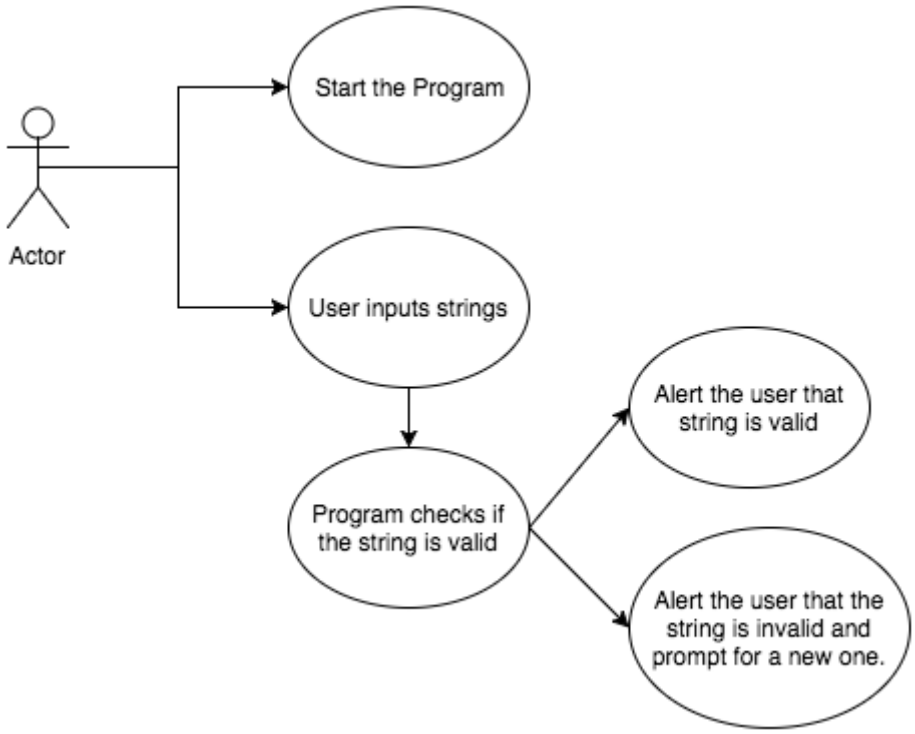
e. Scenario:
   i. Student – Start the program
   ii. Student – Through the command line, call the load function.
   iii. Program – Using the input load information from the command line, retrieve the pre-existing XML file at the given location.
   iv. Student – Can now use the command tool to utilize the loaded XML file.

Use Case 3 Diagram:



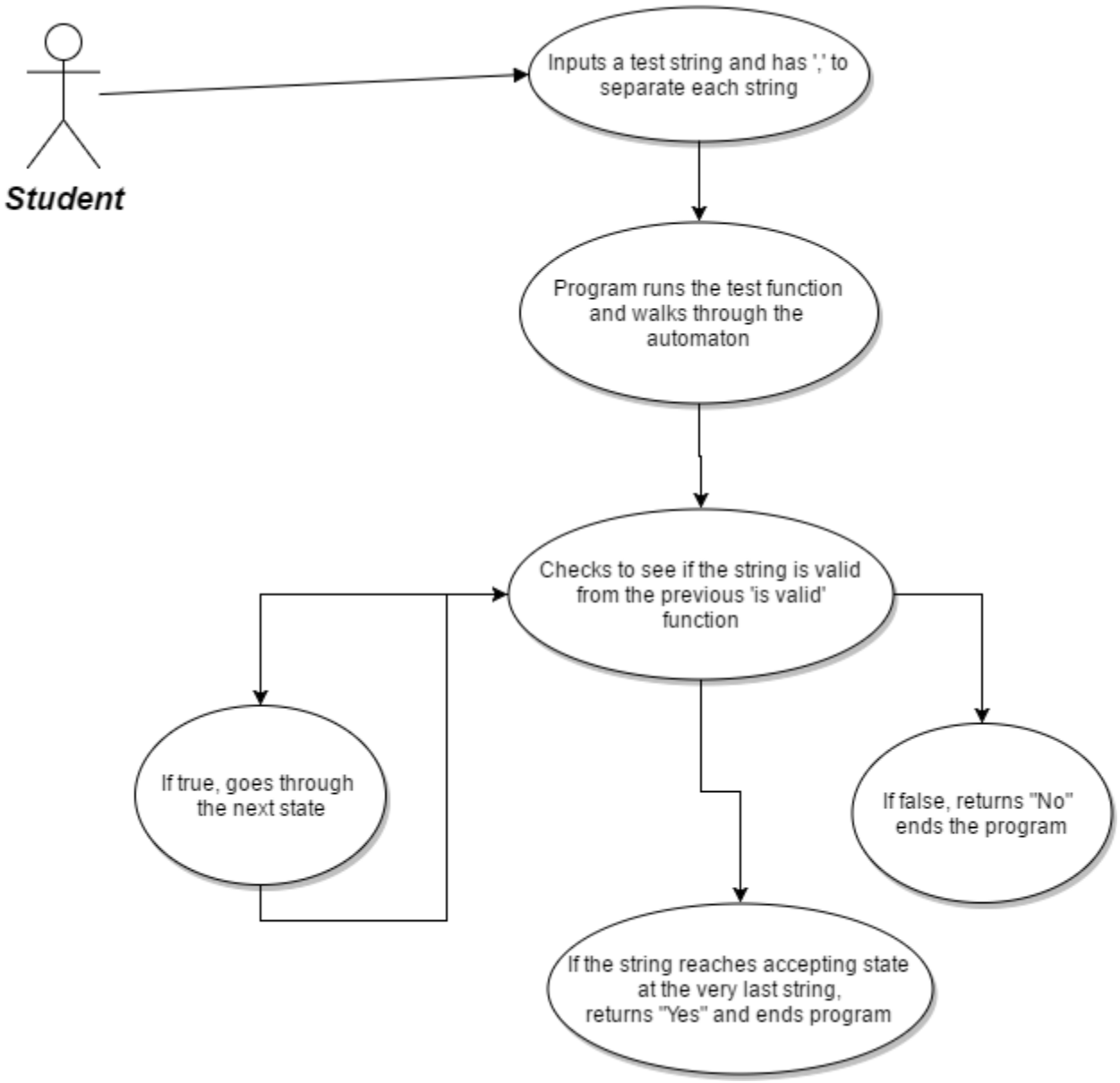
4) Determine if the Given String is Valid
   a. Primary User
      i. Student (Programmer) / Teacher (evaluator)
   b. Goal in Context:
      i. The student (programmer) will input a string, the function will then be able to determine if the string is valid, meaning that the string will be accepted to "walk" to another state in the automata.
   c. Precondition:
      i. A command line function to take user inputted string.
      ii. An already created deterministic finite automata.
   d. Trigger:
      i. The user will input a string into the command line. The program will then trigger the validation check.
      ii. Returns a Boolean Value
   e. Scenario
      i. Student – Start the program.
      ii. Student – Input string into the command line
      iii. Program – Check all possible "walk" options for the automata. If the string is accepted by one of these paths, then notify the user that the string is validated.

Use Case 4 Diagram:



5) Traverse through the Automata using test inputs:
    a. Primary User:
        i. Teacher (evaluator) / Student (Programmer)
    b. Goal in Context:
        i. This function will walk through the given automaton using use case 4. This function will not just see if the string is valid but it will also go through the states and see if the set of string is valid
    c. Precondition:
        i. The tester needs to know what his/her set of test input strings are.
        ii. The program needs the fourth use case done
    d. Trigger:
        i. The tester inputs a set of string and uses a “,” comma to separate each strings. Every single string will call the fourth use case to see if its valid. If valid, go to next string.
    e. Scenario:
        i. Tester – Loads a pre-existing XML file for a finished automaton
        ii. Tester – Inputs: “10,01” in the testing function.
        iii. Program – Notices that the testing function is called and walks through the automaton.
        iv. Program – runs the fourth use case function and determines that ‘01’ is valid.
        v. Program – since ‘01’ is valid, goes through the next state and checks if its valid.
        vi. Program – the state is valid and ends the program.

Use Case Diagram 5:



Summary:

Git Lucky's first milestone was incorrect and we made 5 different test case for automata rather than making 5 use cases. Jomar Dimaculangan (contact person) asked if our milestone 1 was correct and it wasn't. This is a correction of that and Professor Haipeng gave us permission for resubmission. Jomar Dimaculangan contributed by listing out the five use cases and making use case 1 and use case 2. Jomar wrote how our "create an automaton" and the save function will be implemented in the project. Mathew Merrick contributed by making use case 5. Mat listed out how test string inputs will be implemented. Devon Honig created use case 3 and 4 which are loading from an XML file and determining if a single string is valid.