

12/15/2017

Jeremy Roberts
13D Ward Hall
Kansas State University

Dear Prof. Roberts:

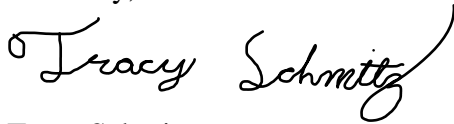
Please find enclosed my revised manuscript, “Development of a Computer Application for Creating Machine State Diagrams Using Python,” which I would like to submit to you as part of the course requirements of ME 701.

This paper describes the design of a standalone computer application that will take in user input about machine states and machine state connections and create a visual that represents interconnectivity.

My code can be found at <https://github.com/tgschmitz/Machine-State-Diagram-Generator.git>

I thank you for your consideration and look forward to your decision.

Sincerely,

A handwritten signature in black ink that reads "Tracy Schmitz". The signature is written in a cursive style with a large, sweeping flourish at the end of the last name.

Tracy Schmitz

Review 1

This document covers the work that the writer performed creating a graphical user interface which creates a graphic. The graphic creation tool was created using PyQt5. Some grammar mistakes are present throughout this paper. Some key points in this document are ambiguous.

1. Grammar/other mistake examples:

1. second sentence of PROGRAM DESCRIPTION “without and easy”

RE: I agree with this comment and I implemented this change.

2. After all the connections checkboxes are selected” should this be connections’?

RE: I agree that this is a little ambiguous. I changed the sentence to read “After all the connection checkboxes are selected.”

2. Ambiguity in section Output Window

By saying “Figure 2 shows the desired outcome from my application”, the document makes the reader have to ask the question, does this application actually produce Figure 2. or is this tool as of yet unfinished and Figure 2. is what the programmer hopes to have their tool create once finished. Try to look through other sections as well, there may be some more ambiguity.

RE: At the time of the draft report I did not have working code and used an image of what I desired my output to look like. With this revision of the report, I have completed-working code and have updated Figure 2. to show the actual output graphical.

Review 2

In the introduction section, it may be helpful to add an example figure or simple machine state diagram. You could then explain that diagram, like how state 3 depends on state 1 and state 4. This may help the reader understand exactly what you mean when you say state diagram, because you could have different terminologies across different disciplines. You will also want to search for some references (textbooks, journal articles) you can cite in work. Typically, most references are used in the introduction (or theory) sections of a paper.

RE: I agree that terminology may vary across disciplines and have incorporated a more detailed description of state diagrams in my introduction.

In Architecture Overview you say, “The Boolean values from the check-boxes are assigned to a matrix as a two or a zero for easy manipulation.” What do you mean by easy manipulation? What are you doing with this data once this matrix is formed? It would be good to explain how you go from this matrix to the arrows connecting the different states on your final image.

RE: Values in a matrix are easier to access when applying them to other functions within python. Once the matrix is formed I can use the Boolean(true-false) values to ascertain whether an arrow is drawn in my graphic. I agree that I need a description of how the matrix is connected to the arrows and have implemented a description in my final revision.

I’m guessing you will rework the last two paragraphs to explain how your line drawing algorithm works and to show the final output. If you have “future work” like maybe your second feature that populates the GUI with the correct number of machine states, list that in a Future Work heading or with your conclusions.

RE: I agree and have incorporated a Future Work heading.

The following comments are mostly grammar/formatting suggestions

Comment 1. Start first page of the actual transcript/paper at 1.

RE: This change has been implemented.

Comment 2. "and" should be "an"

RE: This change has been implemented.

Comment 3. "is" to "its"

RE: This change has been implemented.

Comment 4. Typically, you want to stay in 3rd person for technical writing. Avoid using I, we, me, and you.

RE: These changes have been implemented.

Comment 5. Fig 2. Needs a caption. (I’d change this out with an actual output from your program when you get your code working properly)

RE: This change has been implemented.

DEVELOPMENT OF A COMPUTER APPLICATION FOR CREATING MACHINE STATE DIAGRAMS

Tracy Schmitz

Mechanical Engineering
Kansas State University
Manhattan, Kansas 66506
Email: tgschmitz@ksu.edu

ABSTRACT

This paper covers the basic functionality and architecture of a software application that creates a machine state diagram similar to a flowchart. The program allows a user to input the names of “states” or “conditions” and the connections between them. The graphical user interface consists of a single column of editable text boxes for the states and a matrix of checkboxes to define connections between the states. This application uses python programming language and PyQt5 library to implement a user-friendly interface.

INTRODUCTION

Many engineers across various definitions find themselves programming in a wide variety of different languages and it can be hard to understand code. A machine state diagram has various states as shown in Figure 1. as “A”, “B”, “C”, “D”. The interdependencies are shown with arrows that detail the way that the state can change from one to the other. Figure 1. shows that A can transition to B and C, but B cannot transition to A like C, because the arrow is not bi-directional.

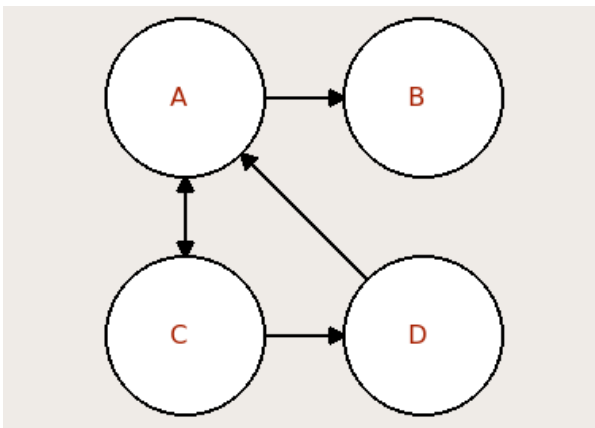


FIGURE 1. MACHINE STATE DIAGRAM EXAMPLE

Machine State Diagrams can be implemented with code to illustrate the behavior of an entity or program as a result from its inputs as well as its preceding states. It can also be used to show how an entity responds to various events that change it from one state to another. This program can be applied to other systems such as electrical hardware that undergo various states with interdependencies. Lastly, the simplest use of this program can be used to create flowcharts describing steps in processes. Various programs exist to create these diagrams but often require the user to draw the diagram which can be a tedious time-consuming process. For this reason, there is a need for an application that would allow the user to specify states and their interdependencies using an efficient and intuitive programming environment. The goal of this project is to create an application that meets these needs using the Python programming language.

Program code is open source located in a repository under MSDG.py at:

<https://github.com/tgschmitz/Machine-State-Diagram-Generator.git>

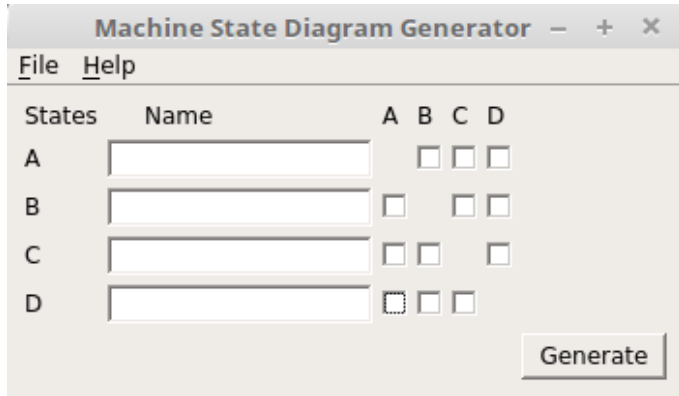
PROGRAM DESCRIPTION

From an end user perspective, the most important part of a computer application is the user interface. Without an easy to use interface, the desired output can become more difficult than it needs to be. The interface needs to be easy and intuitive to produce meaningful results quickly and effectively.

Graphical User Interface (GUI)

The GUI for this program was made simple to provide an intuitive and fast user experience. The interface consists of three areas of user input; an array of vertical line-edits labeled alphabetically, a matrix of checkboxes, and a single button used

to generate a separate task pane displaying the resulting figure. The line edits are where the user will input the desired number of states to be included in the final diagram. Above the matrix of checkboxes, the columns of checkboxes are labeled alphabetically. The user then checks the corresponding box that defines a relationship between the state defined on the left side



of the matrix and is alphabetical symbol above the matrix. After all the connection checkboxes are selected, the generate button at the bottom of the task pane is clicked to generate the output diagram. The output diagram includes boxed states and arrows that define the directional relationships between the states. A sample view of the GUI is shown in Figure 2. below:

FIGURE 2. GRAPHICAL USER INTERFACE

Output Window

The output window contains the diagram consisting of the state boxes and their connections. The desire is to produce an image that can be saved as a .png or .JPEG file that can then be used in other applications. Figure 3. shows the desired outcome from my application.

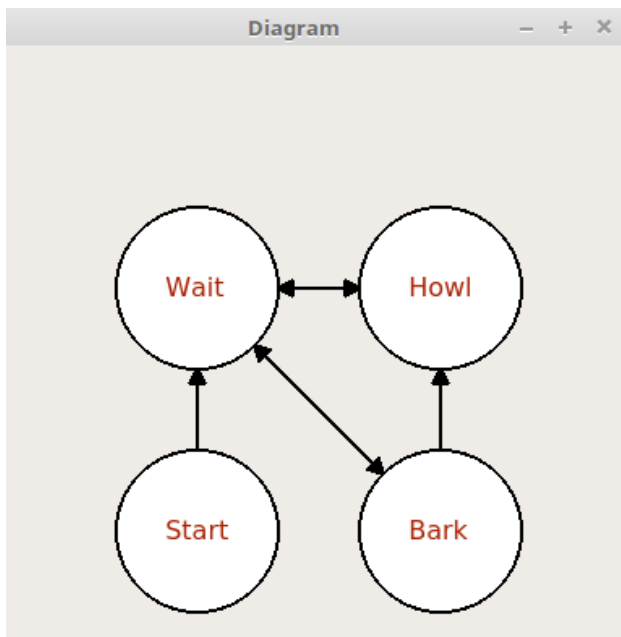


FIGURE 3. STATE DIAGRAM OUTPUT WINDOW

IMPLEMENTATION

This section covers the basic operations of the program, including brief descriptions of modules used and the structure of the source code.

Modules

This program relies primarily on PyQt5. PyQt5 was used to construct all the elements of the GUI and used its class structure as a framework to implement multiple windows and the PyQt.QtGui's QPainter, QPen, QColor, and QBrush modules to create the diagram construction portion of the software.

Architecture Overview

The architecture of this program consists primarily of the GUI but behind the scenes the values that are input by the user are routed using some logic to create the final output image. The information that is input by the user into the line-edit text boxes is routed to each state box drawn using the QPainter module and a second class within the PyQt5 mainWindow. If a line-edit text box is left blank, a for-loop is used to separate it out so that it isn't included in the final diagram. The Boolean values from the check-boxes are assigned to a matrix as a two or a zero for easy manipulation. The values assigned to the matrix are easier to access when applying them to other functions within python. Once the matrix is formed the Boolean(true-false) values can be used to ascertain whether an arrow is drawn in my graphic.

Arrow Lines: PyQt5 layers graphics as they are drawn pixel by pixel which allows certain graphics to be drawn over top of previously drawn graphics. This allowed the arrow lines to be drawn from the center coordinates of the state bubbles and then the bubbles are drawn which covers up the portion of the line needed. This eliminated the need to determine the angle of the arrow line. As the matrix is looped through it compares the row and column of each state, if either one returns a true value the line is drawn connecting the two states.

Arrow Head: The head of the arrow is more complex incorporating a polygon function called createPoly. The function is passed the number of vertices, radius of polygon, angle of intersection, state coordinates, and the distance from the center of the state bubble. The distance from the center of the state bubble and the angle of intersection are converted from polar coordinates to rectangular and added to the center coordinates of the state bubble.

State Bubbles: Next the state bubbles are drawn over the lines and the QBrush module is used to fill the bubbles with white. The text is filled in last to make sure that it shows up correctly and is located using the center coordinates off the state bubbles.

FUTURE WORK

Time has limited this project from being developed further. However, there are a few features that would be easy to incorporate that would improve this program a great deal. The

first being a way to incorporate more states possibly implementing a way for the user to specify the number of states that would be required in the diagram. Then the main window would populate with the correct number of line-edit text boxes and an even matrix of check-boxes for state connections. This could be done with a for loop and another line edit. The second feature would stem from the first because as the number of states increases it becomes more difficult to arrange the state bubbles in an orderly manner so that the number of crossed state lines is minimized.

CONCLUSION

This project presents a fully functional program that meets the need for a tool allowing engineers to create publication ready diagrams for documentation of circuitry, software, operational business models, industrial processes, and any other operational structures and processes that need to be conveyed concisely. However, future development would expand the capabilities as the current version only allows for four states.

ACKNOWLEDGMENTS

A huge thanks to the stack overflow community at <http://stackoverflow.com> for teaching me how to implement several of the features associated with PyQt5 that allowed me to produce meaningful code. I'd also like to acknowledge my sister Tricia Schmitz for her help with general programing questions related to this projec

