Report

This report details how the automatic code generator created by the author functions. The type of automatic code generator the author created is a “Finite State Machine” or “FSM.”

In the Dictionary of Algorithms and Data Structures Paul Black defines an FSM as:

> “A model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state. Computation begins in the start state with an input string. It changes to new states depending on the transition function. There are many variants, for instance, machines having actions (outputs) associated with transitions (Mealy machine) or states (Moore machine), multiple start states, transitions conditioned on no input symbol (a null) or more than one transition for a given symbol and state (nondeterministic finite state machine), one or more states designated as accepting states (recognizer), etc..[1]”

The FSM the author created functions in the following way. First, there is a file which consists of the following data:

s0,i0:s1,i1:s0

s1,i0:s3,i1:s2

s2,i0:s0

s3,X

This data is called a “text file” and is actually a representation of a many more lines of computer code. If the “text file” were shown in its full form it would contain 49 lines of code and be called the “generated code.”

The purpose of the FSM is to change the generated code by altering the text file. Specifically, it is to make a small change to the code of the text file to create a large change to the text of the generated code.

For example, imagine if a user wanted to change this data. Without an FSM multiple lines of computer code in the generated code would have to be altered. To achieve the same change with the FSM only a few tokens need to be altered.

Therefore, an FSM radically increases productivity by reducing the amount time needed to alter data for a different use. If the author’s company were to use the FSM, then it would significantly reduce the amount of time that software programmers needed to adapt existing data for new uses. This in turn would reduce company costs.

---

[1] Paul Black (2019). Finite State Machine. Retrieved from: https://xlinux.nist.gov/dads/HTML/finiteStateMachine.html.

Letter

Dear Bill,

Regarding our conservation about using my Finite State Machine or "FSM" for company purposes I have attached a short report to this letter (see above) explaining how it works.

In addition, within this letter I have put details about the limitations and strengths of the FSM, the feasibility of this project, the availability of similar products and my recommendations about its use.

Limitations and Strength

There are two major limitations to finite state machines. First, they have to be kept updated. As Gabrielle Tomassetti says "A code generator tool must be maintained. If you created it you must keep updating it, if you are just using an existing one you have to hope that somebody keep maintaining it or you have to take over yourself. So, the advantages of code generation are not free. This is especially risky if you do not have or cannot find the right competencies to work on the generator."[2] Therefore although my FSM may reduce the time spend on computer coding by employees there will still need to be some staff time dedicated to keeping the FSM updated.

Secondly, computer code created by finite state machines is much more complex than computer code written by a human computer programmer. In some circumstances this does not matter. But some software programs need to be less complex in order to maximize their effectiveness. Gabrielle Tomassetti summarizes this concept by saying "Generated code is also surely less optimized than the one you can write by hand. Sometimes the difference is small and not significant, but if your application needs to squeeze every bit of performance code generation might not be optimal for you."[3]

There are four major strengths of finite state machines. First as already mentioned in the attached report an FSM can increase productivity by allowing existing data to be used for different purposes by making small changes to the text file and not having make large alterations to the generated code.[4] Secondly, the FSM is generally easier to understand. This is because it is usually easier to read the text file than the generated code.[5]

Also, the FSM can also be repurposed to use different types of code. Regarding this Tomassetti states "Once you have a process to generate code for a certain language or framework you can simply change the generator and target a

---

[2] Gabrielle Tomassetti. (2018). A Guide to Code Generation. Retrieved from: https://tomassetti.me/code-generation/.
[3] Gabrielle Tomassetti. (2018). A Guide to Code Generation. Retrieved from: https://tomassetti.me/code-generation/.
[4] Gabrielle Tomassetti. (2018). A Guide to Code Generation. Retrieved from: https://tomassetti.me/code-generation/.
[5] Gabrielle Tomassetti. (2018). A Guide to Code Generation. Retrieved from: https://tomassetti.me/code-generation/.

different language or framework. You can also target multiple platforms at once. For example, with a parser generator you can get a parser in C#, Java and C++. Another example: you might write a UML diagram and use code generation to create both a skeleton class in C# and the SQL code to create a database for MySQL. So, the same abstract description can be used to generate different kinds of artifacts."[6]

Finally using an FSM means that you consistently get the same code each time you use it. When software is coded by programmers it can result in different lines of code depending on who creates it.[7]

Feasibility

This section of the letter will examine how feasible using an FSM is for our business by examining scientific journal articles regarding the efficacy of automatic code generators. The Apache Velocity program is an automatic code generator that is an open source software tool that is used most often in webpage creation. Tomingas, Kliimask and Tammet detail how Apache Velocity is used to render objects created in Java code in a simpler to use template form.[8] A benefit of this is that it allows web designers to alter the appearance of the website without having to alter as much of the underlying computer code.[9] Sturm, von Voss and Boger detail how Apache Velocity is used to make commonplace functions of a website simpler.[10] For example Apache Velocity is used to automatically generate automatic emails for purposes such as password reminders by storing this information in the text file rather than the storing it directly in Java generated code.[11] Our company could use this tool to build our own or client's websites.

Actifsource is an automatic code generator that is proprietary. Stoffel defines it in a recent journal article as "It's a modeling tool, which allows you to model your problems with UML. Such modeling tools can assist you in areas where there's a lot of boiler plate code and where modeling complex relationships is important,"[12]. There were two journal articles that reviewed the efficacy of Actifsource – one in 2016 and one in 2019.[13] The first article in 2016 details that one of the advantages of Actifsource is that users can write templates without using a special template coding

---

[6] Gabrielle Tomassetti. (2018). A Guide to Code Generation. Retrieved from: https://tomassetti.me/code-generation/.
[7] Gabrielle Tomassetti. (2018). A Guide to Code Generation. Retrieved from: https://tomassetti.me/code-generation/.
[8] Tomingas, K., Kliimask, M., & Tammet, T. (2015). Data integration patterns for data warehouse automation. In New Trends in Database and Information Systems II (pp. 41-55). Springer, Cham.
[9] Tomingas, K., Kliimask, M., & Tammet, T. (2015). Data integration patterns for data warehouse automation. In New Trends in Database and Information Systems II (pp. 41-55). Springer, Cham.
[10] Sturm, T., von Voss, J., & Boger, M. (2002, September). Generating code from UML with velocity templates. In International Conference on the Unified Modeling Language (pp. 150-161). Springer, Berlin, Heidelberg.
[11] Sturm, T., von Voss, J., & Boger, M. (2002, September). Generating code from UML with velocity templates. In International Conference on the Unified Modeling Language (pp. 150-161). Springer, Berlin, Heidelberg.
[12] Stoffel, R. (2010, December). Comparing Language Workbenches. In MSE-seminar: Program Analysis and Transformation, University of Applied Sciences Rapperswil (HSR), Switzerland (pp. 18-24).
[13] Actifsource. (2016). Actifsource Workbench. Retrieved from: https://www.actifsource.com/products/actifsource_workbench/index.html.

language. [14] Another advantage of Actifsource stated in the 2019 article is that it gives real time feedback to the user if they make errors. It does this by validating the templates users create with the backing models.[15] Actifsource could be useful for our business by allowing us to use a graphical editor that is specifically suited to our needs rather than a general use tool.

Availability of Similar Products

There are many existing automatic code generators and finite state machines available for use. These are both commercial products and open source. However, I would recommend that our business either use open source products or my own FSM if it is applicable. The reason I suggest this is because the cost of the commercial FSM can be quite prohibitive. Since our company's purpose in using this technology is to reduce costs it seems counterproductive to buy these products. This is especially true because of the wide availability of high-quality open source products which are free.

Recommendations

I recommend that our company utilizes the FSM that I have created. However, I would like to make one thing clear. My FSM has the potential to increase productivity especially in the area of software programming. However, our company will still need to use software programming to alter the FSM to be used for new data, new coding languages or new uses of existing data. So, while my FSM will reduce company expenditures on software programming it will not eliminate them.

[14] Tkachuk, M., Martinkus, I., Gamzayev, R., & Tkachuk, A. (2016). An integrated approach to evaluation of domain modeling methods and tools for improvement of code reusability in software development. Informatik 2016.

[15] Martinkus, I., Mayr, H. C., Nagornyi, K., & Tkachuk, M. (2019). Evaluation of the Effectiveness of Domain Modeling Methods in Terms of Model Complexity. Archived Volume, 513.