2.1.3.1 Procedural Programming

Return to Programming Paradigm

Procedural Language break a problem down into a small set or procedures (i.e., Functions or Procedures) that when put together can solve the original bigger problem. In addition to the procedure calls there is the control and flow of which procedures need to be called and in which order (control flow). In addition, data can be shared and stored between the invocation of the procedures. A very important, simplistic used to visualize the activities in Procedural programming is the **Flow Chart**. However, the number and quantities of graphical tools used to represent many procedural processes is much greater than the simple Flowchart. For more details, see UML/SysML/SOAML as well as UPDM[1].

- **Note:** Flowcharts are NOT required to when representing programs or applications BUT a flowcharting technique can be used to help explain a program or an application, than it is most likely is using a Procedural Paradigm.

Some examples of Procedural Programming languages are:

- Ada
- C/C++
- C#
- ECMAScript (i.e., Javascript)
- FORTRAN
- Java
- Pascal
- Python

In Figure 1, the green and red dots represent the starting and ending points respectively, the green boxes represent procedures, the diamonds represent a decision point and the blue cylinders represent the data. Note there alternates paths through the application.

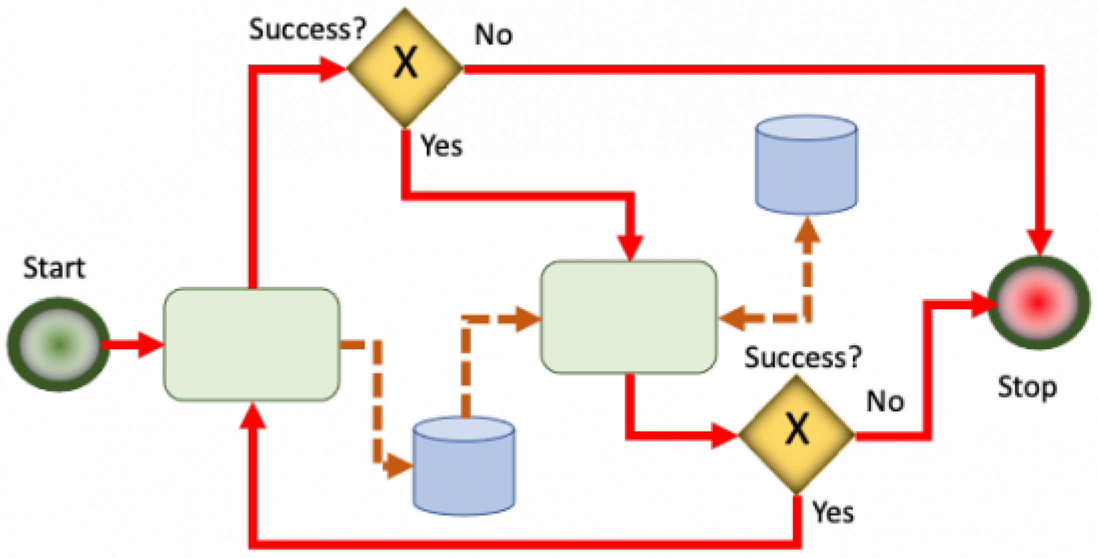

Figure 1: Procedural Programming Control Flow

1) Bleakley, Graham, Introduction to UPDM, Accessed: 13 April 2021, https://www.omg.org/ocsmp/Introduction_to_UPDM.pdf