Fault injection techniques are widely used to assess dependability and fault handling capabilities of software systems.

High level fault injection can be used to understand the impact due to errors in program elements 
- local variables, function arguments, return addresses. These elements determine the data flow in the application

NFTAPE is a general purpose configurable tool for automated fault injection based dependability assessment. Its basic components are Target Generator, Control Host, and Injector.

**An Enhancement to NFTAPE for High Level Fault Injection in C++ Applications**

Varsha Chittawar, Indian Institute of Technology, Kharagpur
ADVISORS: R. K. Iyer, Zbigniew Kalbarczyk, Nithin Nakka

**Background**

- Fault injection techniques are widely used to assess dependability and fault handling capabilities of software systems.
- High level fault injection can be used to understand the impact due to errors in program elements
  - local variables, function arguments, return addresses. These elements determine the data flow in the application
- NFTAPE is a general purpose configurable tool for automated fault injection based dependability assessment. Its basic components are Target Generator, Control Host, and Injector.

**Research Plan**

- Understand the working and methodology applied in development of NFTAPE.
- Study the differences in the structure of debug information for C and C++ files.
- Accordingly modify existing algorithm in NFTAPE to generate breakpoint addresses and offsets for local variables in C++ applications.
- Automate the injector by linking modified target generator with injector to inject faults.
- Modify control host graphical user interface to support the target generator and injector.

**Goals**

- Developing fault injector for C++ applications as an enhancement to NFTAPE framework. This includes:
  - Developing target generator for local variables in C++ applications.
  - Link modified target generator with injector to accomplish fault injection.
  - Modify control host to comply with new target generator and injector.

**Fundamental Questions/Challenges**

- Identifying the differences between structure of debug information for C and C++ programs.
- Traversal of tree-like representation of debug information of C++ program for determining offset of local variables.

**Algorithm to determine offsets of local variables.**

**Research Results**

- Modifying existing algorithm to support generation of breakpoint addresses and offsets for local variables of C++ applications.
  - Changes made keeping in view salient features of C++. Specific cases studied:
    - function overloading
    - function overriding
    - inline functions
    - class member functions.
- Interfacing target generator with injector for command line execution of injector.
- Demonstration of implemented features on test applications.