**Explicación de la Solución - Bryan Pérez – Dylan Torres**

**I. BOSQUEJO DEL PROBLEMA:**

Primero que todo, vamos a describir lo que queremos que nuestro sistema lleve a cabo:

**1)** Recibir Un conjunto de entradas (S, E, r, sigma, T y M) ya sea por medio del componente GUI de la interfaz de usuario o por medio de un archivo txt

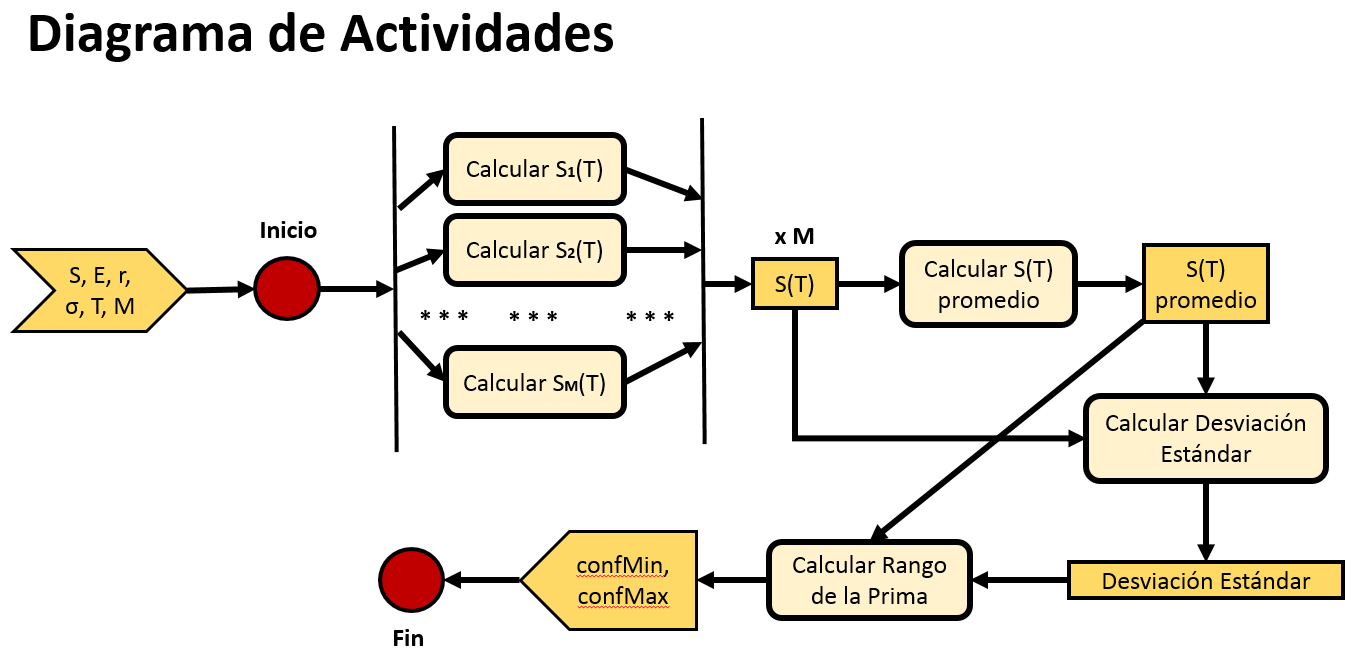
**2)** Con base en ese grupo de entradas, se simulan M valores **S(T),** de tal forma que tengan una distribución normal.

**3)** Cuando los M cálculos anteriores hayan terminado, calculamos el S(T) **promedio**.

**4)** Con el conjunto de todos S(T) simulados y el S(T) promedio, generamos la **desviación** **estándar**.

**5)** Con la desviación estándar y el S(T) promedio, obtenemos el **Rango** de la **Prima**, representado en dos valores (el límite **Mínimo** (confMin) y el límite **Máximo** (confMax))

**6)** Se lanzan esos valores, o por **pantalla** o se guardan en un **archivo** .txt (Según sea lo deseado por el usuario)

Este proceso se ve mejor reflejado en el siguiente gráfico

**II. SOLUCION PROPUESTA:**

**A) Diagramas de Deployment: Visión General Del Sistema:**

El sistema completo se estructura como lo muestra el diagrama siguiente:

\*\*Ver archivo adjunto – DEPLOY PROYECTO\*\*

**B) Componentes Generales: Visión Específica Del Sistema:**

Ahora entremos en detalle sobre los tres tipos de nodos que conforman el sistema completo.

**1. Servidor:**

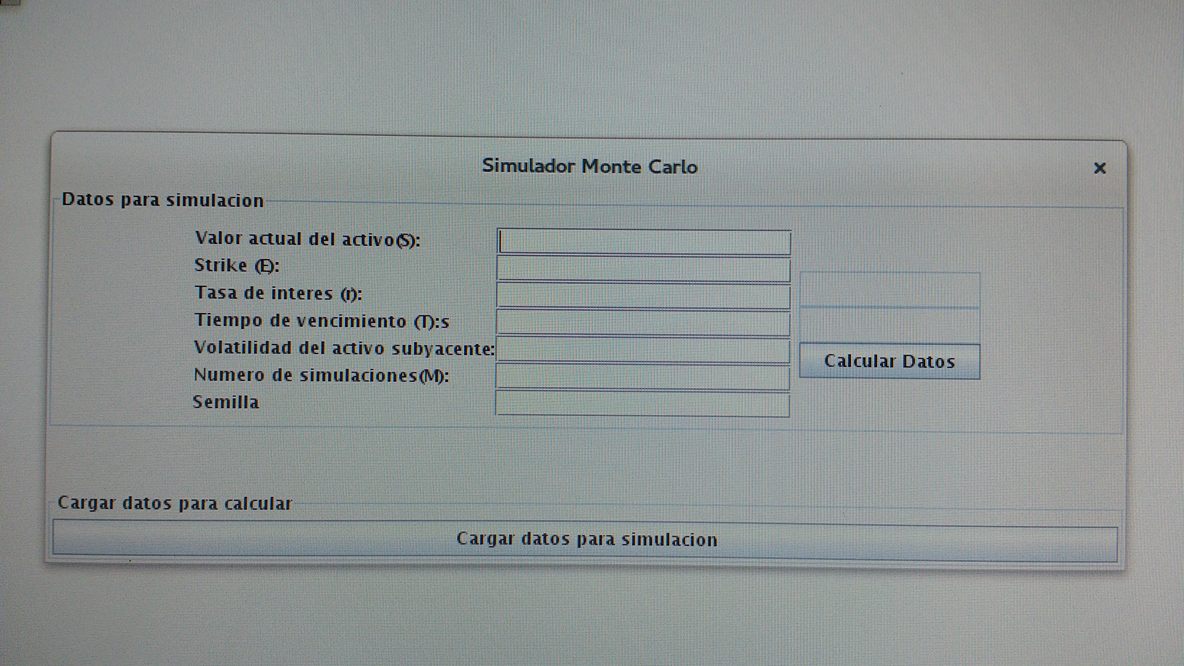
**Responsabilidades**:

* Hacer los cálculos finales de un call, una vez se hayan recibido las M simulaciones que este necesita
* Exportar los diferentes resultados de los call, ya sea en el GUI de la interfaz gráfica o en un archivo de texto plano .txt
* Enviar una semilla al Generador de Números Aleatorios por medio del Protocolo TCP
* Guardar la Información de los diferentes call que se tienen que procesar
* Hacer un balance de carga para determinar cuántas simulaciones se le va a pedir a un determinado cliente
* Recolectar los datos de entrada por medio de dos maneras:

- Por componente GUI (Interfaz Gráfica)

- Por Medio de un archivo de Texto Plano.

Para satisfacer estos dos requerimientos, generamos la siguiente interfaz gráfica:



**2. Cliente:**

**Responsabilidades:**

* + Generar un conjunto de simulaciones S(T) para luego enviárselos a los clientes
  + Pedirle al Generador un conjunto de números aleatorios para poder crear las simulaciones anteriores
  + Preguntarle al Servidor periódicamente si necesita ayuda para simular algunos S(T)

**3. Generador de Números Aleatorios:**

**Responsabilidades:**

* Actualizar su objeto Random con una semilla cada vez que se lo solicitan
* Generar un paquete de números aleatorios que cumplan con una distribución normal y enviárselos al cliente que los pidió

**B) Estructura de distribución en hardware:**

La estructura que estamos trabajando para el proyecto es la tipo **SIMD** (Single Instruction Multiple Data).

Nuestro sistema se conforma por varios **esclavos** que llevan a cabo la misma tarea, en este caso dichos esclavos son los nodos clientes los cuales le envían un grupo de simulaciones al servidor.

En una estructura tipo SIMD también hay un **orquestador** y en nuestro caso ese sería el Servidor, el cual va recibiendo todas las simulaciones que los diversos clientes le van pidiendo y las va guardando y administrando.

**C) Estructura de memoria:**

El tipo de sistema que vamos a manejar es una estructura tipo **UMA**, Uniform Memory Access.

Todos datos (en este caso las simulaciones S(T) ) van a residir dentro del nodo donde se está ejecutando el servidor y ahí es donde se van a procesar finalmente

Lo que hacen los demás nodos es enviar información hacia el Servidor para que este los vaya almacenando y los procese cuando sea necesario.

**D) Comportamiento entre Componentes:**

\*\*\* Ver Archivo Anexo en Power Point “Protocolos TCP - Comportamiento” \*\*\*