Cálculos Probabilísticos
Software RMES

Centro de Desarrollo de Gestión Empresarial.
1 Oriente 1097 - Viña del Mar, Chile.
Fono:(56) (32)688987 - Fax:(56) (32)2684079
empresa@mes.cl

Abril, 2010

Resumen

El presente informe tiene como objetivo explicar el proceso de cálculo para los indicadores probabilísticos del software RMES y proponer una solución para mejorar su rapidez.

Índice

1. Introducción

Todo desarrollo informático tiene como base un diseño o un esquema, símil a los planos de un arquitecto, que da horizonte a los programadores.

RMES contiene muchos diseños según sea la tarea que se quiera realizar: almacenar datos, calcular, mostrar resultados en pantalla, etc. Uno de estos diseños (el más básico, medular e importante de RMES) es el utilizado para modelar el concepto de planta en un sentido sistémico, que básicamente fue creado para soportar la implementación de cálculos probabilísticos, entre otras cosas. Este diseño será conocido como el **Árbol de Subsistemas**.

Su forma, como cualquier diseño informático, es un meta-lenguaje por sobre la especifidad de la aplicación que se dé. Es así como su comprensión es natural para cualquier individuo con conocimientos de Ingeniería en Mantención.

El **Árbol de Subsistemas** implementado en RMES no es difícil de entender, pero antes, es necesario comprender 2 conceptos importantes dentro del área de desorrollo informático orientado a objetos:

- Interfaz: es la definición del comportamiento de "algo", el qué se debe hacer. El concepto es tan abstracto que se puede llevar a cualquier área del conocimiento que se asocie con comunicación.
- Clase: es una definición de cómo hacer las cosas. Es la encargada de llevar dentro de ella la lógica de algo.

Llevando estos conceptos al área de mantención, se puede ver que dentro de una planta existen varias máquinas o áreas, que a su vez engloban a otras máquinas o áreas. Claramente, esta es la definición básica de un árbol. Por ende podemos definir los siguientes conceptos:

- MachineCollector: es una interfaz que define, entre otras cosas, el comportamiento de algo que contiene hijos.
- Chain: es una clase que implementa el comportamiento MachineCollector, por lo tanto, puede contener hijos. Ademas, es el concecpto que define a la raíz del árbol (por efectos de programación, en un árbol siempre es necesario conocer y definir una raíz)
- Subsystem: es una clase que tambien implementa el comportamiento de una MachineCollector, por lo tanto, puede contener hijos pero que no es una raiz. He aqui la diferencia con la clase anterior.

Pero también sabemos que dentro de la planta existen varios tipos de máquinas: en serie, en paralelo, etc. Por lo tanto, aquello implica crear clases para cada tipo. Es aqui donde nace el concepto de **Herencia**, los distintos tipos de máquina deben heredar las propiedades de un Subsystem dado que todos ellos son subsistemas. Es asi como se crean las siguientes clases:

- Equipment: representa una máquina.
- LinearMachine: representa un subsistema en línea.
- ParallelMachine: representa un subsistema en paralelo.
- RedundantMachine: representa un subsistema en redundancia.
- FractionMachine: representa un subsistema en fraccionamiento.
- StandbyMachine: representa un subsistema en Stand By.

Existen otras clases e interfaces que por simplicidad no se explicarán. Esta es la base del modelo implementado en RMES para una planta .

2. El proceso de cálculo probabilístico

La lógica anteriormente explicada implica que debe existir una interfaz que defina el comportamiento de un Subsystem respecto a los cálculos que debe realizar (dado que es posible que la forma de calcular un indicador difiera entre cada tipo de subsistema, lo cual es así en algunos casos). Esta interfaz esta "camuflada" dentro de la clase Subsystem y define 4 métodos importantes:

1. getAvailability() (A): retorna disponibilidad de un subsistema.
2. getMtbf() (MTBF): retorna el mtbf de un subsistema.
3. getMttr() (MTTR): retorna el mttr de un subsistema.
4. getLambda() (λ): retorna la tasa de fallas de un subsistema.

La implementación de cada uno de estos métodos (funciones) se realiza en cada tipo de subsistemas, pero generalmente se puede encontrar lo siguiente en casi todos los tipos:

$$A = \frac{MTFB}{MTBF + MTTR}$$

$$MTBF = \left(\sum_{i=1}^{n} \lambda_i \right)^{-1}$$

$$MTTR = MTBF \cdot \left(\sum_{i=1}^{n} MTTR_i \cdot \lambda_i \right)$$

$$\lambda = \frac{1}{MTBF}$$

donde $MTTR_i$ y λ_i se refieren a los indicadores asociados a los hijos del subsistema actual. Por supuesto, al llegar a un nivel los valores se obtiene de forma directa. Reescribiendo la fórmula de disponibilidad en términos de $MTTR_i$ y λ_i:

$$A = \frac{\left(\sum_{i=1}^{n} \lambda_i \right)^{-1}}{\left(\sum_{i=1}^{n} \lambda_i \right)^{-1} + \left(\sum_{i=1}^{n} \lambda_i \right)^{-1} \cdot \left(\sum_{i=1}^{n} MTTR_i \cdot \lambda_i \right)}$$

Se puede observar que el término

$$\left(\sum_{i=1}^{n} \lambda_i \right)^{-1}$$

se recalcula 3 veces, más veces aún el término λ_i. Incluso sólo observando a este nivel se puede ver claramente el problema de recálculo que posee la estructura de **Árbol de Subsistemas**, lo cual es un defecto que ocurre en casos particulares como éste.

El problema crece exponencialmente debido a que $lambda_i$ se vuelve a calcular de la misma forma en caso de que los hijos también sean subsistemas. El cálculo es **recursivo**.

Para los casos de Fraccionamiento y Redundancia se utiliza la siguiente fórmula:

$$MTBF = \int_0^\infty R(t)dt$$

donde $R(t)$ es la confiabilidad en el tiempo t del subsistema. No existe una expresión matemática para esta integral, debido a que $R(t)$ depende del subsistema y de sus hijos, y de los hijos de los hijos, y así sucesivamente.

La forma de calcular esta integral es mediante integración numérica. Para ello se utiliza el Método de Simpson el cual es una Integral de Rienmann mejorada. En sí, es ir calculando porciones de áreas una y otra vez hasta llegar a un cierto límite. Este proceso es altamente costoso dado que se debe recalcular muchas veces el valor de $R(t)$ debido a que t varía.

El Método de Simpson para calcular integrales es uno de los más efectivos y rápidos que existen, por no decir el mejor, por ende, el cuello de botella no se encuentra en cómo hacer la integral, si no en el proceso obtener el cuerpo de ella, los valores de $R(t)$.

3. Soluciones

Una solución es implementar un caché orientado al cálculo almacenando los resultados parciales que se van obteniendo. Ejemplificando con la fórmula de disponibilidad:

$$A = \frac{\left(\sum_{i=1}^{n} \lambda_i\right)^{-1}}{\left(\sum_{i=1}^{n} \lambda_i\right)^{-1} + \left(\sum_{i=1}^{n} \lambda_i\right)^{-1} \cdot \left(\sum_{i=1}^{n} MTTR_i \cdot \lambda_i\right)}$$

Almacenar

$$MTBF^* = \left(\sum_{i=1}^{n} \lambda_i\right)^{-1}$$

Luego

$$A = \frac{MTBF^*}{MTBF^* + MTBF^* \cdot \left(\sum_{i=1}^{n} MTTR_i \cdot \lambda_i\right)}$$

A pesar de que el ejemplo es burdo, es útil para comprender la idea a esta escala.

Otra solución es revisar la implementación de todos los cálculos probabilísticos en pos de buscar simplificaciones como la siguiente:

$$A = \frac{\left(\sum_{i=1}^{n} \lambda_i\right)^{-1}}{\left(\sum_{i=1}^{n} \lambda_i\right)^{-1} \left(1 + \left(\sum_{i=1}^{n} MTTR_i \cdot \lambda_i\right)\right)}$$

$$A = \frac{1}{1 + \left(\sum_{i=1}^{n} MTTR_i \cdot \lambda_i\right)}$$

Una última mejora es implementar caché a nivel de resultado.

Es neceario crear casos de prueba que permitan realizar las modificaciones con seguridad para cualquier solución que se elija.