# Algoritmo Histórico

*Software RMES*

Centro de Desarrollo de Gestión Empresarial
1 Oriente 1007 – Viña del Mar – Chile
Fono: (56) (32) 2688987 – Fax: (56) (32) 2684079
empresa@cgessa.com

Esteban Heidke Adriasola
Consultor
René González Leiva
Ingeniero de Desarrollo
Marzo 2011

René González Leiva
Ingeniero de Desarrollo
Septiembre 2012

**Resumen**

El presente informe tiene como objetivo documentar el algoritmo histórico implementado en RMES para el procesamiento sistémico de fallas explicando los todos procesos involucrados de forma que puedan ser entendidos y evaluados por los expertos en mantención. Este documento no esta dedicado a explicar la obtención Indicadores de Confiabilidad, sino a detallar los conceptos básicos que permiten obtener cualquiera de ello. Si ya se conocen todos los procedimientos descritos aquí, se recomienda remitirse al documento (Cuevas, 2012). Este documento requiere que el lector tenga conocimientos básicos en el uso y conceptos asociados al software RMES y la ingeniería en confiabilidad. Además, se necesita que el lector este familiarizado con el documento (Gonzalez, 2012).


Contenido

# Ilustraciones



# Tablas


# Introducción

El documento (Gonzalez, 2012) introduce los términos principales que utilizará este documento. Por ende, se dará por hecho de que el lector comprende los siguientes conceptos:

1. **Nodo**
2. **Bloque**
3. **Configuración**
4. **Tupla**

Entendiendo estas definiciones, es posible saber como afecta el no funcionamiento de los nodos hijos sobre los nodos padres, idea recursiva que comienza desde los bloques hacia los nodos padres llegando hasta el nodo raíz, o sea, la planta o flota completa.

Gracias el esquema RBD y el uso de los conceptos de nodo y tupla es posible obtener KPIs históricos a todo nivel de la planta y de cualquiera de sus componentes.

Existen muchas configuraciones lógico-funcionales en las que se puede encasillar a una configuración (en serie, paralelo, fraccionamiento, entre otras). Éstas rigen el comportamiento de cómo deben ser tratadas las tuplas que llegan desde los nodos hijos hacia los nodos padres.

Para efectos en la explicación de algunos temas del documento, se adelantará que existen al menos 4 tipos de fallas:

1. MP (Mantención Preventiva Programada)
2. MC (Mantención Correctiva No Programada)
3. DO (Detención Operacional Programada)
4. DONP (Detención Operacional No Programada)

Las MC de dividen en subtipos y existen muchos mas tipos de fallas, pero por ahora no nos interesan y nos bastará solo con estas cuatro.

El algoritmo histórico utiliza el concepto de recorrido Post-Orden, explicado en (Gonzalez, 2012). Se recomienda comprenderlo antes de continuar con este documento.

El algoritmo histórico se divide en 2 grandes partes:

1. Algoritmo de Propagación Histórica
2. Algoritmo de Cálculo Histórico

Las secciones Propagación Histórica y Calculo Histórico detallarán ambos algoritmos respectivamente. La sección Métrica explicará ciertas consideraciones que se realizan para obtener indicadores de confiabilidad.

# 1. Propagación Histórica

## 1.1.Falla Histórica

El algoritmo histórico internamente no utiliza las tuplas para realizar los cálculos dado que contiene información que no es útil, y el hecho de incluirla en el algoritmo provocaría que fuese más lento, utilizara más recursos del sistema (computador) y contuviera información que nunca se usaría. Por ello, se utiliza una reducción conocida como **Falla Histórica**. En este contexto, se debe comprender este concepto como un **no funcionamiento genérico de un nodo**, el cual puede deberse a diferentes razones (tipo de falla) y no a una razón en particular.

### 1.1.1. Falla Unitaria

La **Falla Unitaria** es un tipo de **Falla Histórica**. Es una simplificación de una **tupla**, la cual sólo conserva la información justa y precisa que permita la propagación hacia niveles superiores del árbol de nodos. Los campos de una **Falla unitaria** son:

- Fecha Original
- Fecha Inicial y Final
- Tipo de Falla
- Impacto

Al iniciar el algoritmo de propagación histórica, sobre todo nodo se genera una duplicación y transformación de información: a partir del conjunto de tuplas que tenga un nodo, se crea un conjunto similar de fallas unitarias obteniendo los campos listados anteriormente, y excluyendo otros, como son el Costo, Síntoma, Modo de Falla, Causa y Orden.

En necesario dejar claro que la **Fecha Original de la tupla** inicialmente es el mismo valor que la **Fecha de Inicio** de la tupla. A pesar de que esto suena redundante, no lo es. Su utilidad se explicará en detalle posteriormente y radica en la implementación de criterios de ordenamiento.

Estos conceptos son la base del algoritmo de propagación. Como ya se explicó anteriormente, la **falla histórica** (y en este contexto, **falla unitaria**) es simplemente una copia de una **Tupla**, pero más simple y con la información justa y precisa.

### 1.1.2. Falla Compuesta

Una **Falla Compuesta** también es una **Falla Histórica**. Su particularidad es que agrupa un conjunto de otras fallas históricas (ya sean unitarias o compuestas) que comparten el mismo intervalo de tiempo. Posee los mismos campos que la Falla Unitario exceptuando el impacto, el cual será la sumatoria recursiva de los impactos de las fallas que contenga.

Suena algo extraño esta definición recursiva de falla compuesta. Su sustento es puramente informático, dado que el algoritmo trata uniformemente fallas unitarias y compuestas como fallas históricas generando procesos particulares para cada una de ellas si fuese necesario.

La sumatoria de impactos del conjunto de fallas de una falla compuesta es menor o igual a 100%. Además, la fecha original de una falla compuesta será la fecha original más anterior de entre todas las fallas que posea.

Como criterio del algoritmo de propagación, el proceso analizado de propagación dejará en todo nodo Redundante, incluyendo al Bloque, fallas unitarias. Las configuraciones en Serie o Fraccionadas podrán contener fallas compuestas y que sólo contendrán fallas unitarias.

Esto será explicado en las próximas secciones.

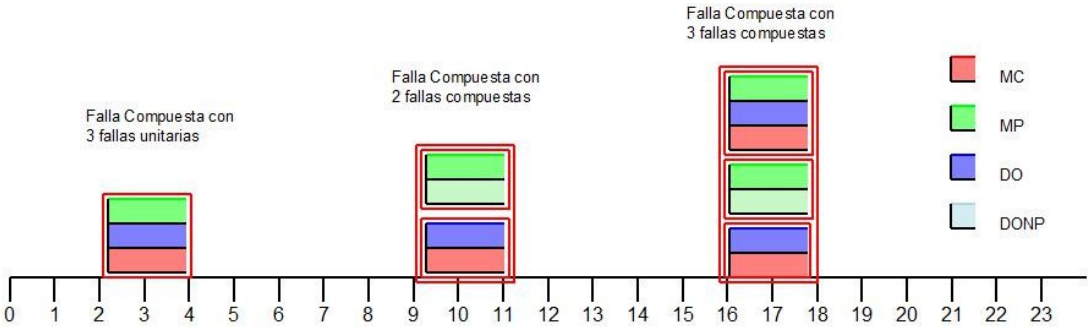


Ilustración 1. Ejemplo de fallas compuestas

## 1.2.Procesamiento de Bloques

El Algoritmo de Propagación Histórica a nivel de bloques es bastante simple, pero necesita que las tuplas en los nodos estén ordenadas y no se encuentran traslapadas, a lo más, pueden estar una inmediatamente de la otra, pero no "unas sobre otras". En otras palabras, de todo el conjunto de tuplas que el bloque contenga, no existe intersección entre 2 detenciones, cualesquiera sean.

Existe un algoritmo que asegura la correctitud de las tuplas en todo momento y se conoce como **Filtro de Intersección** y se aplica cada vez que un nodo sufre cambios es su conjunto de tuplas. Este algoritmo se encuentra explicado detalladamente en el documento (Gonzalez, 2012).

El algoritmo a este nivel simplemente transforma las tuplas a fallas históricas **copiando** las propiedades que se necesitan de las primeras sobre las últimas. El impacto de estas fallas históricas siempre será 1. Como se explicó anteriormente, dado que el usuario selecciona un intervalo de tiempo, solamente se transforman aquellas tuplas que intersecten con el periodo seleccionado. Para las tuplas de los extremos, se generan fallas obteniendo únicamente el intervalo de intersección entre la tupla y el periodo de estudio (en realidad, esto es lo que se realiza para todas las tuplas del bloque, se transforma a falla histórica solo aquellas tuplas que generan un intervalo de intersección mayor a cero).

Es importante dejar en claro que la transformación de tupla a falla histórica implica una **duplicación** de información, de esta forma, **la tupla jamás es modificada por el algoritmo histórico.** Teniendo claro esto, podemos notar que el algoritmo histórico finalmente trabajará con copias de la información original, además de procesar una cantidad masiva de datos.

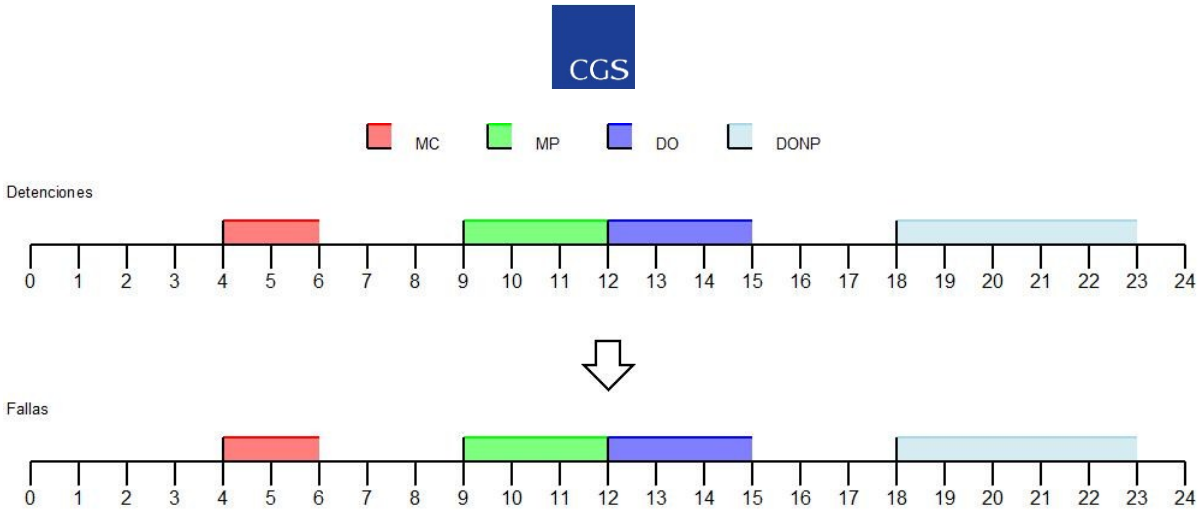


Ilustración 2. Procesamiento y duplicación de fallas históricas a nivel de bloque

## 1.3.Procesamiento de Configuraciones

A nivel de configuraciones, cualquiera sea su configuración lógico funcional, es necesario realizar 3 subprocesos básicos:

1. Procesar las tuplas que contenga la configuración: proceso símil al realizado sobre un bloque, las tuplas están ordenadas y no traslapadas (o sea, se aplicó anteriormente el Filtro de Intersección), siempre tendrán impacto igual a 1 y solo se trabaja con aquellas que estén dentro del intervalo de tiempo seleccionado por el usuario.
2. Procesar las fallas que tengan los nodos hijos: este proceso también es simple. Las fallas de los hijos ya vienen listas para ser usadas, por ende, solo **se copian** directamente hacia el padre (se replica la información del hijo sobre el padre), modificando los impactos de las fallas según el impacto del hijo que provenga (simplemente multiplicando ambos impactos), lo cual sólo se aplica a configuraciones en Fraccionamiento, como se explicó en las secciones anteriores.
3. Procesar en conjunto las fallas del padre y de los hijos: este proceso es algo más elaborado, por lo que se explicará a continuación. Su objetivo es obtener finalmente el impacto de las fallas históricas del todo sobre la configuración. Este proceso se divide en 2 partes: el **Guillotinado** y la **Jerarquización**.

### 1.3.1. Guillotinado

El proceso de **Guillotinado** agrupa el conjunto completo de fallas padre/hijos en una sola línea de tiempo (es muy similar a lo que ocurre con el popular juego Tetris, en donde se van apilando bloques unos sobre otros; en nuestro caso, solo tendríamos bloques horizontales de largo variable, que representarían nuestras fallas históricas, sobre una línea temporal).

Luego, se realizan cortes por inicio y fin de todas las fallas, generando más fallas de duración menor. Es posible imaginar esto como utilizar una enorme guillotina con incontables cuchillas y utilizarla sobre el conjunto completo de fallas históricas. Las cuchillas estaría alineadas según los

inicios y fin de cada fallas histórica en el conjunto. Si una fecha se repite, basta con una cuchilla que represente a la fecha.

Lo anterior provoca que cada falla histórica se divida en otras varias más. Por supuesto es posible que existan algunas que no se dividan, lo cual no afecta en absoluto al algoritmo.

Lo importante de lo anterior es que **toda falla que se dividió en otro pequeño conjunto de fallas, comparten la misma fecha original**. Esto es muy importante dado que se usa como criterio de ordenamiento y jerarquización.

Observando la Ilustración 3 (leer desde abajo hacia arriba), la primera falla MCM de la configuración S, se dividió en 2 fallas de duración distinta, con intervalos distintos, pero con la misma fecha original.

Se reitera que las fallas de los hijos en este punto se encuentran corregidas (los impactos de las fallas de los hijos fueron corregidos según el tipo de configuración, lo que técnicamente, se aplica solo en caso de configuraciones en fraccionamiento, como se explicó anteriormente). Luego de este subproceso, se continúa con el procesamiento particular según sea el tipo de configuración de la configuración. El estudio se realiza intervalo por intervalo respecto a los generados por el subproceso anterior.

Por ejemplo, observando la Ilustración 3, se obtiene los siguientes intervalos:

- De 9:00 hrs a 11:00 hrs.
- Intervalos de 1 hora desde las 11:00 hrs a las 19:00 hrs.

Por cada intervalo se eligen las fallas que finalmente quedarán sobre la configuración. **Si alguna de estas fallas pertenece directamente la configuración, esta será la única que prevalecerá.**

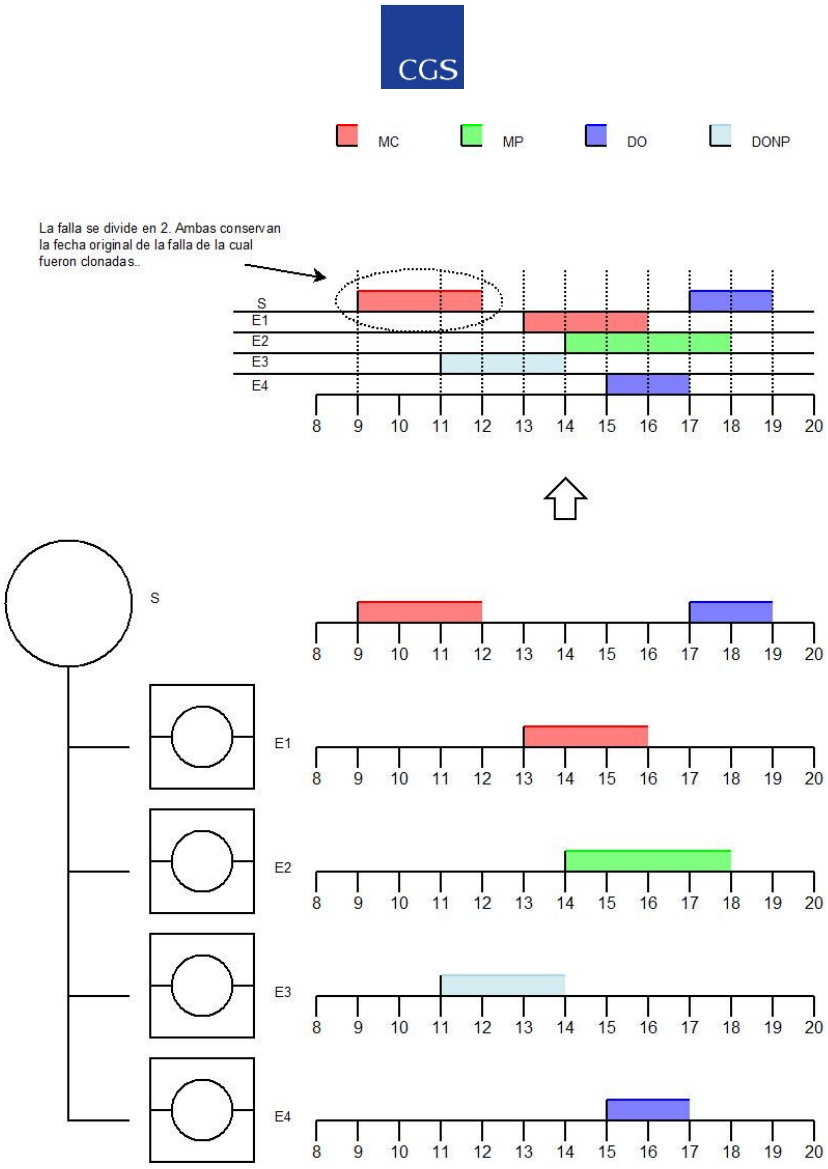


Ilustración 3. Subproceso de Guillotinado

### 1.3.2. Jerarquización

#### *1.3.2.1. Configuraciones Redundantes*

El algoritmo para procesar configuraciones en Redundancia Parcial, en Paralelo o en Standby es el mismo. Cada una de estas configuraciones define un valor informático conocido como "Fail Break Count" (FBC), que se refiere a la cantidad de fallas que la configuración soporta hasta de dejar de funcionar, es decir, si la cantidad de fallas en un intervalo de tiempo es igual o supera este valor, la configuración se detiene.

Sea una configuración Redundante de tamaño *n*, si se encuentra en paralelo soporta *n - 1* fallas, por ende, su valor FBC será de *n*. El valor para una configuración en StandBy es 2. Para una configuración en Redundancia Parcial (n:m), el valor FBC es (n - m + 1). Por ejemplo, para un (4:3), el FBC es 2.

La Ilustración 4 representa el conjunto completo de fallas que han subido una Redundancia Parcial (4:3) (leer desde abajo hacia arriba). A la configuración sólo sube una falla unitaria por cada intervalo, y es aquella que hace detener a la configuración. Las fallas asociadas directamente a la configuración prevalecen siempre. En nuestro caso, el valor FBC es 2. La fecha original de la falla es importante para configuraciones Redundantes. Por ejemplo, si se observa el intervalo entre las 13:00 hrs y 14:00 hrs, existe una falla MCM perteneciente al bloque E1 y una falla DONP perteneciente al bloque E3. Dado que esta última falla es la que inicia primero, la culpable de que en ese intervalo de tiempo la configuración en Redundancia Parcial falle es la falla MCM del bloque E1, por lo tanto, es la falla que sube a la configuración en Redundancia Parcial.

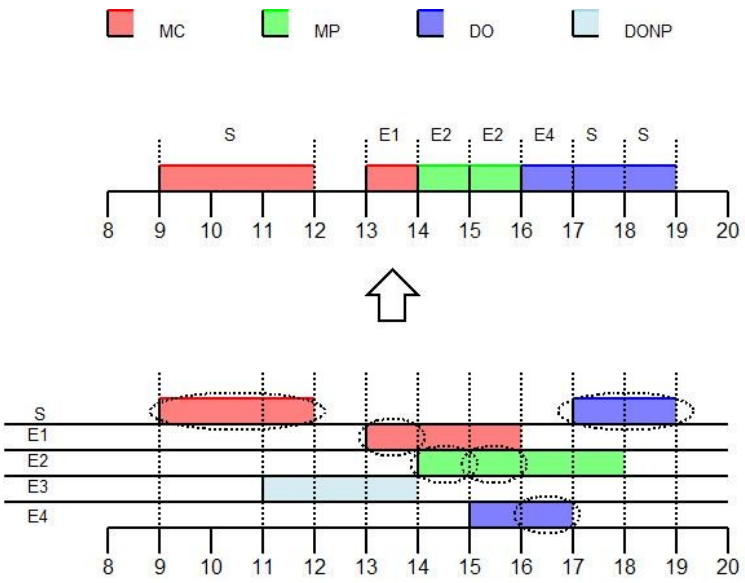


Ilustración 4. Jerarquización para configuración redundante (4:3)

### 1.3.2.2. *Configuraciones en Línea*

Las configuraciones en línea tiene la particularidad de que fallan si es que alguno de sus nodos hijos lo hace. Dado que esta configuración soporta fallas fraccionadas, el criterio de general de jerarquización es el siguiente: del conjunto de fallas existentes en un intervalo, se elige la de mayor impacto, en caso de que hubiesen varias, se elige la que haya comenzado antes (menor fecha original).

La Ilustración 5 muestra el modifica el ejemplo general de forma que las fallas en cada bloque tengan un impacto asociado cambiando los bloques por fraccionamientos, para soporte de fallas fraccionadas y, para no generar confusión, los impactos no están asociados a las configuraciones fraccionados, si no que a las fallas que subieron en ese momento particular (las fallas son las que tienen el impacto, no los fraccionamientos. Recordar que no tiene sentido de que los hijos de una configuración en serie tenga impacto).

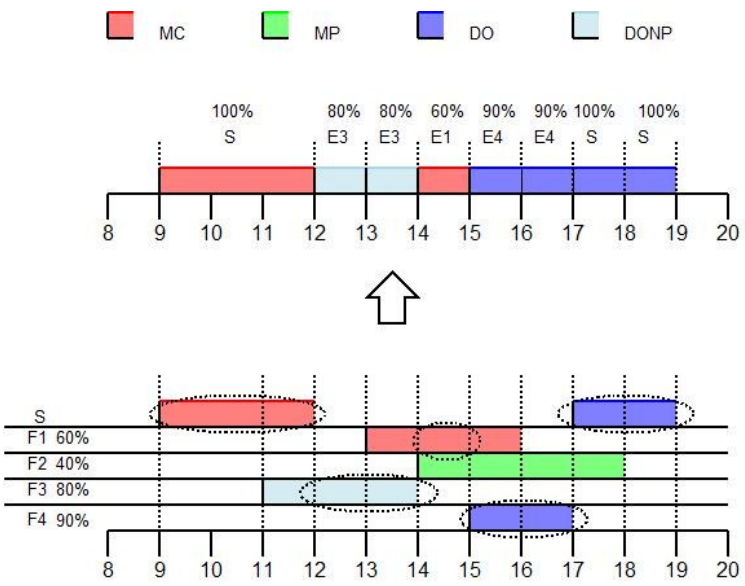


Ilustración 5. Jerarquización para configuraciones en línea

#### 1.3.2.3. *Configuraciones en Fraccionamiento*

A pesar de que existen 3 tipos de modelos distintos en RMES para Fraccionamiento, el algoritmo histórico pude ser resumido en uno solo. En primer lugar, se busca la sobrecapacidad (OC) de la configuración que es en cuánto más el nodo soporta su no caída. Esto se obtiene de la sumatoria de impactos menos 100%:

$$OC = \sum_i I_i - 1$$

Donde $I_i$ es el impacto de los nodos hijos de la configuración en Fraccionamiento.

La sobrecapacidad de un fraccionamiento simple es 0 (cero, no posee sobrecapacidad), la de uno con capacidad ociosa no supera al impacto mínimo del conjunto de nodos hijos y la de uno con redundancia *(n, m)* será $\frac{n}{m} - 1$.

A continuación se describe el proceso matemático, que llamaremos **Algoritmo de Sobrecapacidad**, que impide que el impacto final de un conjunto de fallas supere el 100%, en caso de que la sobrecapacidad sea mayor que cero (este proceso es necesario debido a que una configuración a lo mas puede fallar en un 100%).

Supongamos que tenemos k fallas que suben a la configuración en un intervalo de tiempo. Cada falla posee un impacto $IF_i$. El **Algoritmo de Sobrecapacidad** corrige los impactos de estas fallas realizando una ponderación según los impactos individuales:

1. Se obtiene la sumatoria original de impactos de las fallas en el intervalo $IF_t = \sum IF_i$
2. Se obtiene la sumatoria corregida de impactos $IF_t' = \sum IF_i - OC$

3. Se obtiene la tasa *r* de ponderación original para cada falla $r = IF_i/IF_t \quad \forall i = 1, \dots, k$
4. Se asigna el nuevo impacto a las fallas $IF_i'' = r * IF_t'$

La Ilustración 6 muestra esta corrección gráficamente. La Ilustración 7 muestra el proceso para un fraccionamiento simple. La Ilustración 8 muestra el proceso para un fraccionamiento con capacidad ociosa.

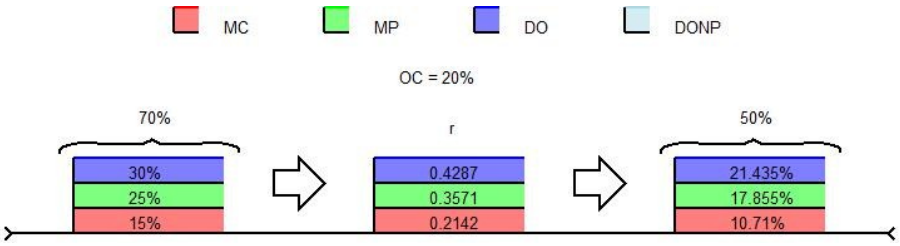


Ilustración 6. Corrección de Sobrecapacidad

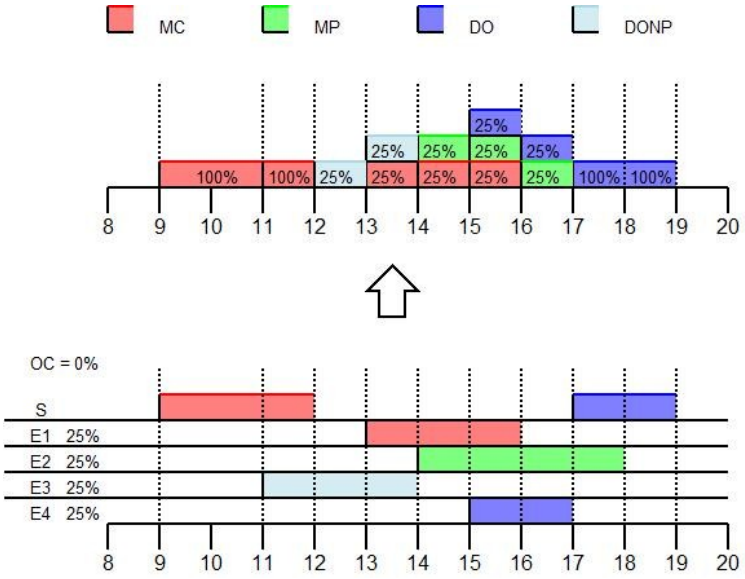


Ilustración 7. Jerarquización para Fraccionamiento Simple

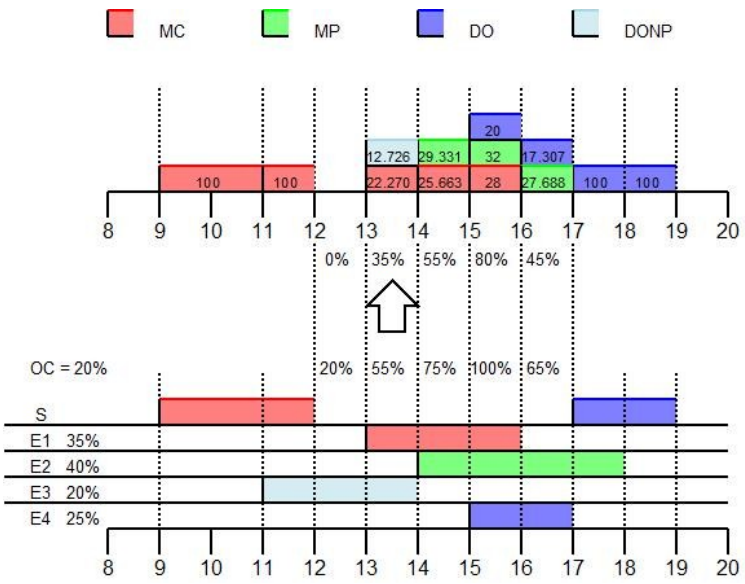


Ilustración 8. Jerarquización para Fraccionamiento con Capacidad Ociosa

En el caso de que a la configuración en fraccionamiento hayan subido varias fallas del mismo tipo, el algoritmo las une en una gran falla del mismo tipo sumando sus impactos. La Ilustración 9 grafica el proceso de unión de fallas.

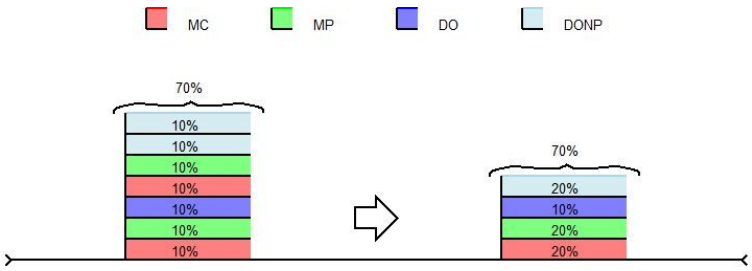


Ilustración 9. Unión de Fallas

## 1.4.Resultado Final

Como ya es de esperarse en vista a los ejemplos de las secciones anteriores, el resultado final del algoritmo de propagación es un conjunto de fallas unitarias y/o compuestas, donde estas fallas unitarias, conjunto que no tendrá un tamaño superior a 8. Esto debido a que las configuraciones en fraccionamientos son los únicos capaces de crear fallas compuestas, y a su vez, unen fallas del mismo tipo en una sola, dejando a lo mas 8 posibles fallas (la cantidad de tipos de falla) presentes en una falla compuesta.




SÓLO PARA USO INTERNO DE CGS

# 2. Calculo Histórico

## 2.1.Proceso de cálculo

El proceso de propagación histórica siempre se realiza para cierto **Intervalo de Propagación**. A su vez, el proceso de cálculo también necesita de un intervalo, el cual puede diferir del anterior. Este se conoce como **Intervalo de Estudio** y define el periodo donde se obtendrán los KPIs. Por supuesto, si los intervalos de propagación y de estudio no se intersectan, los indicadores que se generen no tendrán información relevante.

El Calculo Histórico se realiza individualmente por nodo. Se analizan todas las fallas que la propagación histórica generó sobre este nodo dentro un intervalo de estudio. Podríamos imaginarnos este proceso como un gran contador que finalmente nos dará información básica del comportamiento del nodo como la cantidad de fallas históricas por tipo de falla (MC, MP, etc.), los tiempos asociados por tipo de falla, etc., y además indicadores de confiabilidad tales como disponibilidad, tiempo de reparación y varios más.

## 2.2.Evento

El concepto de **evento** surge para intentar conceptualizar la idea de tupla sobre un nodo, en particular, sobre una configuración.

A nivel de bloque, es relativamente fácil obtener KPIs históricos, dado que las tuplas son objetos de conforman una unidad 'absoluta' de información. Por definición, existen en un periodo de tiempo claramente definido y posee un tipo de falla inequívoco, entre otras propiedades.

A nivel de configuración el esquema cambia radicalmente, dado que es posible que este nivel las fallas históricas que hayan subido desde los hijos se mezclen en un mismo intervalo de tiempo (si se desea, se puede observar la Ilustración 8, el caso de la propagación de fallas de un fraccionamiento con capacidad ociosa, en donde se ve claramente que las fallas históricas que subieron poseen distintos tipos e impactos).

El **evento** agrupa un conjunto de fallas históricas contiguas, como muestra la Ilustración 10.

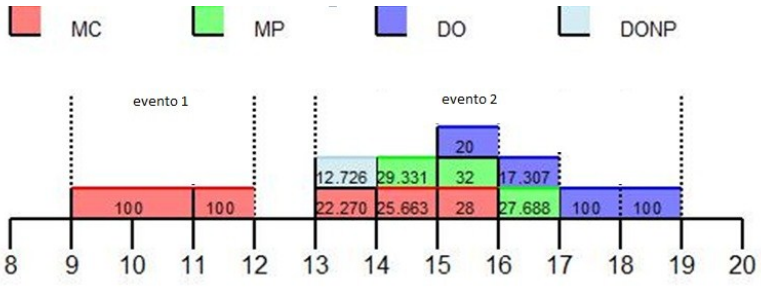


Ilustración 10. Dos eventos, cada uno con una cantidad variable de fallas históricas

La Ilustración 10 conceptualiza mejor la idea de eventos. En base a la definición de evento, se desprenden dos corolarios:

1. Nunca 2 eventos compartirán fallas históricas, o en otras palabras, los eventos de un nodo nunca presentaran traslapes.
2. A nivel de bloque, existirán tantos eventos como fallas históricas.

Como se puede observar, el evento a nivel de configuración cumple una función similar a la falla histórica a nivel de bloque, en el sentido que ambos son objetos no *traslapables*. Recordar que las tuplas en el bloque se copian y transforman a fallas históricas. Dado que las tuplas **nunca** se encuentran traslapadas (existe un algoritmo que en caso de que el usuario ingrese tuplas traslapadas, automáticamente, en base a criterios definidos, las tuplas se modifiquen de forma que no presenten traslapes), las fallas históricas emergentes de ellas tampoco lo estarán, ergo, los eventos tampoco presentarán traslapes). Finalmente, a este nivel obtendremos una lista de tuplas, una lista de fallas históricas y una lista de eventos que representan exactamente lo mismo.

Como se dijo anteriormente, el concepto de evento no tiene mucha utilidad a nivel de bloque, pero es indispensable para realizar los conteos de cantidades y tiempos por tipo de falla a nivel de configuración.

## 2.3.Indicadores por evento

En primer lugar, el evento posee una **Duración Total**, que es simplemente el intervalo de tiempo absoluto que abarca. Por ejemplo, si observamos la Ilustración 10, la duración total del evento 1 es de 4 horas y la duración total del evento 2 es de 6 horas.

Además, para cada evento se pueden desprender un conjunto de indicadores simples **por cada tipo de falla** definido por RMES (MC, MP, DO, etc):

1. **Cantidad**
2. **Duración Equivalente**
3. **Impacto Equivalente**

La cantidad $C_x$ para el tipo de falla 'x' en un evento se define como:

- 0, si para un tipo de falla no existen fallas históricas del tipo 'x'.
- 1, si existe al menos una falla del tipo 'x'.

La Duración Equivalente $D_{eqv\_x}$ para el tipo de falla 'x' es la sumatoria de todas las fallas del evento de tipo x multiplicado por su impacto respectivo:

$$D_{eqv\_x} = \sum D_{i\_x} * I_{i\_X} \quad (6)$$

En donde $D_{i\_x}$ es la duración de la falla histórica de tipo 'x' e $I_{i\_x}$ es el impacto para aquella falla.

Para contextualizar las definiciones de cantidad y duración equivalente observemos la Ilustración 11, la cual presenta 3 casos para un evento que tiene como duración total 4 horas:

- El primer caso tiene solo una falla de tipo MC con impacto 0.2
- El segundo caso tiene 2 fallas de tipo MC, ambas con impacto 0.2
- El tercer caso tiene 4 fallas de tipo MC, todas con impacto 0.2.

Para los 3 casos la cantidad de eventos de tipo MC es 1. A su vez, la duración equivalente para el tipo MC no varía y es 0.80 horas.

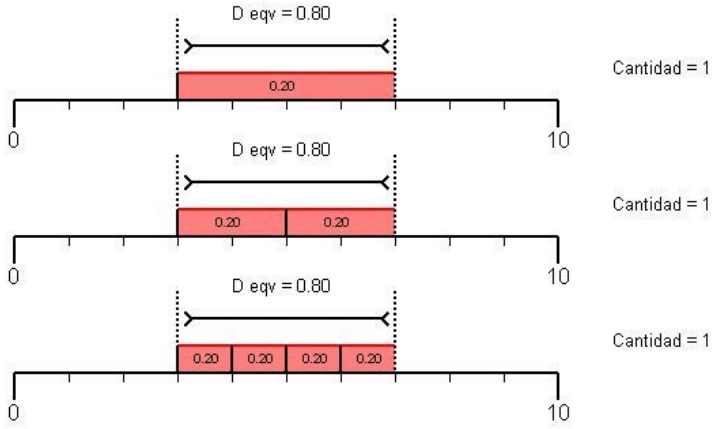


Ilustración 11. Duración Equivalente y Cantidad para un evento con fallas históricas de tipo MC

El Impacto Equivalente $I_{eqv\_x}$ para un tipo de falla 'x' dentro de un evento se define como:

$$I_{eqv\_x} = \frac{D_{eqv\_x}}{D} \tag{7}$$

En donde D es la Duración Total del evento. Entendiendo un poco este concepto, lo que se busca es obtener un indicador del impacto que debería tener una única falla de tipo 'x' pero de duración D. O sea, es como si dentro del evento mezcláramos todas las fallas de un mismo tipo 'x', formáramos una 'masa' amorfa y luego la estiráramos a lo largo del evento generando una sola falla de duración D. Luego, esa falla tendría un impacto $I_{eqv\_x}$.

Para aclarar un poco las ideas, observemos el ejemplo de la Ilustración 12, la cual contiene 3 eventos y cada uno tiene una cantidad variable de fallas con tipos de fallas distintos. La Tabla 1, Tabla 2 y Tabla 3 en conjunto resumen los resultados para cantidad, duración equivalente e impacto equivalente para los tipo de falla MC, MP, DO y DONP.

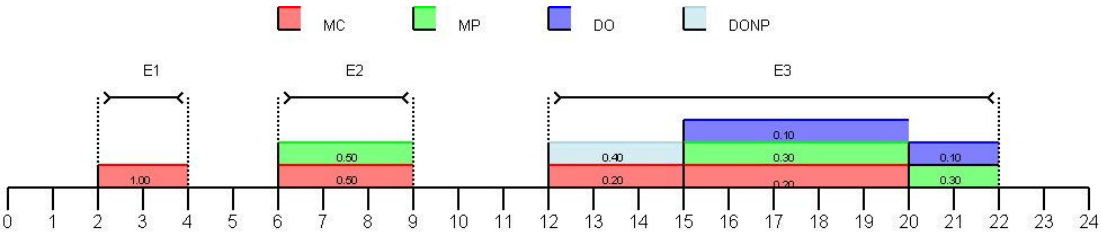


Ilustración 12. Ejemplos de eventos

| Tipo | Cantidad | Dur. Equiv. [h] | Imp. Equiv. |
|---|---|---|---|
| **MC** | 1 | 2,00 | 1,00 |
| **MP** | 0 | 0,00 | 0,00 |
| **DO** | 0 | 0,00 | 0,00 |
| **DONP** | 0 | 0,00 | 0,00 |

Tabla 1. Resumen Evento E1 de duración $D_{E1} = 2$

| Tipo | Cantidad | Dur. Equiv. [h] | Imp. Equiv. |
|---|---|---|---|
| **MC** | 1 | 1,50 | 0,50 |
| **MP** | 1 | 1,50 | 0,50 |
| **DO** | 0 | 0,00 | 0,00 |
| **DONP** | 0 | 0,00 | 0,00 |

Tabla 2. Resumen Evento E2 de duración $D_{E2} = 3$

| Tipo | Cantidad | Dur. Equiv. [h] | Imp. Equiv. |
|---|---|---|---|
| **MC** | 1 | 1,60 | 0,16 |
| **MP** | 1 | 2,10 | 0,21 |
| **DO** | 1 | 0,70 | 0,07 |
| **DONP** | 1 | 1,20 | 0,12 |

Tabla 3. Resumen Evento E3 de duración $D_{E3} = 10$

Volviendo al Impacto Equivalente, la muestra una visión distinta y más cercana a lo que se explicaba anteriormente del significado del indicador. En base a la Ilustración 13, se generan "pisos" por cada tipo de falla y se alargan a la duración del evento contenedor. **Esta transformación es sólo conceptual, no se aplica como procedimiento dentro del algoritmo**. Este apartado solo pretende aclarar un poco más el concepto.

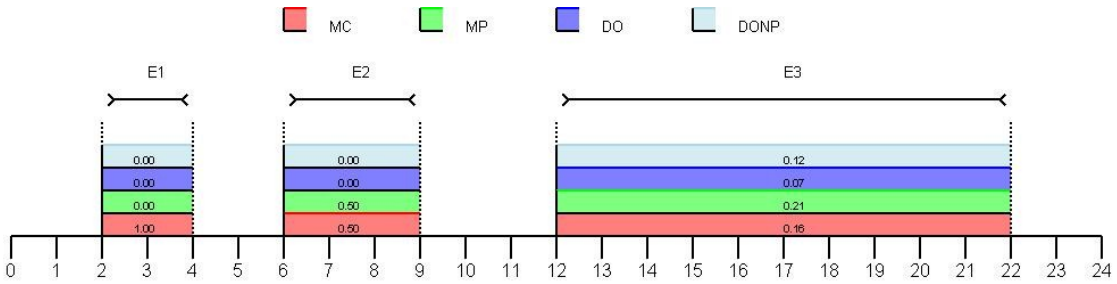


Ilustración 13. Evento como pisos de falla por tipo

## 2.4.Indicadores Básicos

Para cada nodo, lo más probable es que existan un conjunto de eventos, tal como se muestra en la Ilustración 12. Como se vio anteriormente, cada evento contiene resultados parciales por cada tipo de falla. El paso siguiente del algoritmo es obtener indicadores para todo el conjunto de eventos por tipo de fallas. Internamente en el software, esto se conoce como 'Basic Indicators' o **Indicadores Básicos**. Estos se dividen en 3 indicadores por tipo de falla. Por ejemplo, para un tipo de falla 'x' tendríamos:

- **Cantidad total**: sumatoria de la cantidad para el tipo de falla 'x' de cada evento del conjunto.
- **Duración Equivalente total**: sumatoria de la duración equivalente para el tipo de falla 'x' de cada evento del conjunto.
- **Categorización Total**: clasificación del impacto equivalente para el tipo de falla 'x' en base a los siguientes rangos:
  - Entre ]0 %,25 %[
  - Entre [25 %,50 %[
  - Entre [50 %,75 %[
  - Entre [75 %,100 %[
  - Iguales a 100 %

Ejemplificando con el mismo caso de la Ilustración 12. Ejemplos de eventos, nuestros Indicadores Básicos para el conjunto de eventos serían:

| Tipo | Cantidad Total |
|---|---|
| **MC** | 3 |
| **MP** | 2 |
| **DO** | 1 |
| **DONP** | 1 |

Tabla 4. Cantidad Total

| Tipo | Duración Equivalente Total [h] |
|---|---|
| **MC** | 5,10 |
| **MP** | 3,60 |
| **DO** | 0,70 |
| **DONP** | 1,20 |

Tabla 5. Duración Equivalente Total

| Tipo | ]0%, 25%[ | ]25%, 50%[ | ]50%, 75%[ | 100% |
|---|---|---|---|---|
| **MC** | 1 | 0 | 1 | 1 |
| **MP** | 1 | 0 | 1 | 0 |
| **DO** | 1 | 0 | 0 | 0 |
| **DONP** | 1 | 0 | 0 | 0 |

Tabla 6. Categorización Total

## 2.5.Resultado Final

El resultado final del Cálculo Histórico se presenta en las tablas 4, 5 y 6. Recordar que los parámetros del algoritmo son el nodo y el intervalo de estudio. RMES automatiza el cálculo para toda la planta de forma de obtener Indicadores Básicos para todo nodo existente. Ahora sabemos que por cada nodo, se realiza el procedimiento anteriormente descrito.


# 3. Métrica

## 3.1.Indicadores de confiabilidad

La métrica define los indicadores de confiabilidad que finalmente se utilizarán para realizar los estudios respectivos sobre una planta/flota. Todos estos indicadores de confiabilidad se obtienen en base a los Indicadores Básicos obtenidos sobre el par nodo e intervalo.

Dentro de RMES, los indicadores de confiabilidad se dividen en 2 tipos:

1. **Indicadores de Tasa**: Disponibilidad, Utilización y Utilización Efectiva.
2. **Indicadores de Tiempos Medios**: MTBF, MTTR, etc.

Se debe dejar en claro que los Indicadores de Tasa son **únicamente** aquellos que se nombraron, mientras que los Indicadores de Tiempos Medios en realidad son 'el resto de los indicadores' que una métrica contenga. Esta división permite flexibilidad al algoritmo que se explicará en la próxima sección.

Este documento no explicará ningún indicador de confiabilidad debido a que su implementación varía según indicaciones del usuario final. Más detalles en (Cuevas, 2012).

## 3.2.Parámetros

Para que el algoritmo histórico sea más flexible respecto a los distintos requerimientos del usuario final, se cuenta con 2 parámetros:

1. **Tolerancia para duración**: filtro de fallas según su duración.
2. **Corrección hacia la izquierda**: modifica el intervalo de propagación y calculo.

Es en este punto en donde la clasificación en Indicadores de Tasa y de Tiempos Medios cobra sentido. Los Indicadores de Tasa **siempre** se obtienen de lo propagado y calculado en el intervalo en que el usuario selecciono sin ningún tipo de modificación. Los Indicadores de Tiempos Medios **se calculan en base a los parámetros configurados por el usuario**.

Es por ello que dentro de RMES existen 2 conjuntos de fallas por cada nodo: los asociados a la propagación para calcular Indicadores de Tasa y los asociados a la propagación para calcular Indicadores de Tiempos Medios. Ambos conjuntos se diferenciaran en la cantidad de fallas que contenga y el tamaño del intervalo de propagación.

La **Tolerancia para duración** se expresa en horas y es bastante simple de entender. Por defecto este valor es 0, pero cuando se utiliza, su valor es típicamente 0.3333 (20 minutos). Por supuesto, el valor puede ser cualquier otro. Como ya se dijo anteriormente, el algoritmo histórico nunca modifica las tuplas originales de los nodos, por ende, esta tolerancia se aplica en el momento en que se está realizando la duplicación y transformación de tuplas a fallas históricas (específicamente, fallas unitarias) filtrando aquellas tuplas con duración menor a la tolerancia.

La **Corrección hacia la izquierda** es algo más esquivo de entender. **Se refiere a 'mover hacia atrás' el inicio de intervalo de cálculo hacia el final del evento mecánico más próximo anterior al intervalo** (todo esto, en momento de ejecución del algoritmo de cálculo histórico). La Ilustración 14 muestra gráficamente el concepto.

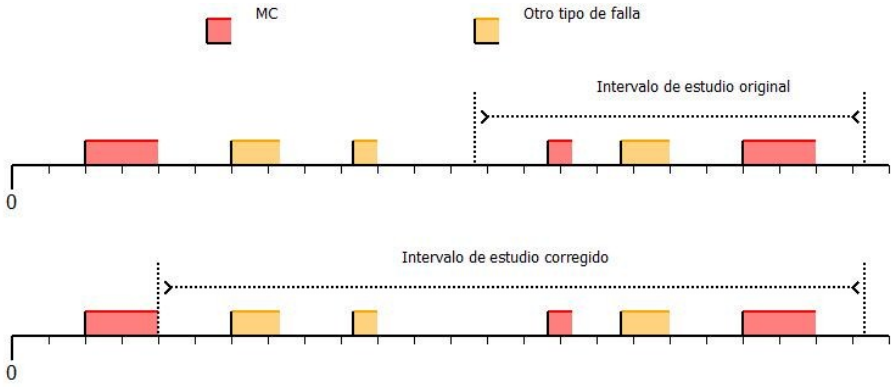


**Ilustración 14. Corrección hacia la izquierda**

Debido al efecto de 'alargar' el intervalo de cálculo (o intervalo de estudio según la Ilustración 14), es posible que se introduzcan al cálculo fallas que no existían en el intervalo original.

La **Corrección hacia la izquierda** se configura mediante un valor booleano (si o no). Cuando está activado, el algoritmo de Propagación Histórica, para todo nodo en la planta/flota, propaga desde la primera falla hasta el fin del intervalo de propagación configurado por el usuario. Recordemos que el intervalo de cálculo existe después del intervalo de propagación. **Si no propagamos las fallas que están antes del intervalo de cálculo, el algoritmo de cálculo no las verá**.

Luego de haber propagado de esta forma, es el algoritmo de Cálculo Histórico el encargado de buscar la fecha final asociada al evento MC. Si no se encuentra ningún evento, el intervalo de cálculo no se corrige. La Ilustración 15 presenta un ejemplo más elaborado en donde se muestran varios criterios más fácilmente explicables con una imagen.

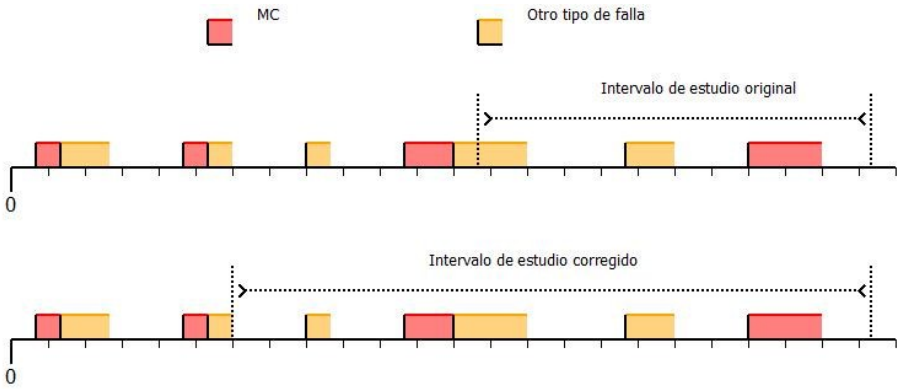


**Ilustración 15. Caso complejo de corrección hacia la izquierda. Se busca la fecha final del evento MC más próximo por la izquierda. El evento que interseca al intervalo de estudio no se toma en cuenta.**


Finalmente, se obtiene los Indicadores Básicos desde esta fecha en adelante, hasta el fin del intervalo original.

## 3.3.Resultado Final

Recapitulando:

- En cada nodo existirán 2 conjuntos de eventos.
- El primero se obtiene de aplicar el Algoritmo Histórico sobre la planta/flota sin tipo de modificación. Con esto se obtendrán los Indicadores de Tasa, los cuales se calculan de igual manera para todos los Flavors.
- El segundo se obtiene de aplicar el Algoritmo Histórico sobre la planta/flota utilizando los parámetros de **Tolerancia para duración** y **Corrección hacia la izquierda**, en caso de que estén configurados. De aquí se obtendrán el resto de los indicadores que se implementen en RMES.

Internamente, el proceso en RMES no es exactamente como el descrito, dado que se utilizando otras ideas, que no vienen al caso explicar, que aumentan la performace del algoritmo, pero conceptualmente, los pasos anteriores son correctos.

# Bibliografía

Cuevas, P. (Septiembre de 2012). *Manual Técnico de los Reporte Software R-MES.*

Gonzalez, R. (Septiembre de 2012). *Introducción a Algoritmos en RMES.*