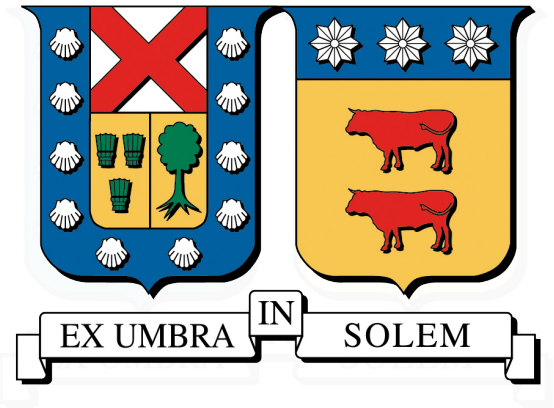
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INDUSTRIAS
VALPARAÍSO- CHILE



MODELAMIENTO DINÁMICO DE CONFIABILIDAD Y PROPUESTA METODOLÓGICA

YASNA ZAMORANO HAMMAD

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INDUSTRIAL

PROFESOR GUÍA : SR. FREDY KRISTJANPOLLER
PROFESOR CORREFERENTE : SR. TOMAS GRUBESSICH

4 de Diciembre de 2016

Índice de Contenidos

1 | INTRODUCCIÓN

La confiabilidad de sistemas tiene por objetivo la creación de un modelo que represente el tiempo para fallar del sistema completo, lo que permite su evaluación y la posterior toma de decisiones sobre diseño. Por ello un modelamiento adecuado significa mejores decisiones.

En la teoría clásica los modelos estáticos de confiabilidad de sistemas ofrecen una forma fácil para obtener la evaluación de confiabilidad de un sistema a partir de un modelamiento. Sin embargo, con el tiempo se han hecho latentes las limitaciones de éste tipo de modelos (estáticos) por al menos dos razones: incapacidad para modelar comportamientos más realistas del funcionamiento de un sistema y la utilización del supuesto de independencia estocástica de las componentes que implica que una componente no puede intervenir en el funcionamiento de otra.

Ante las limitantes de los modelos estáticos existe un avance en el estudio por modelos dinámicos que ha llevado al nacimiento de Dynamic Fault Tree (DFT) y el modelo Dynamic Reliability Block Diagram (DRBD), entre otros. Y que dan lugar a preguntarse cómo efectivamente aplicar modelamiento dinámico de confiabilidad.

En el presente trabajo se estudiará una forma de modelamiento dinámico de confiabilidad de sistemas: el modelo DRBD, posterior a ello se realizará una propuesta metodológica para aplicar el modelo estudiado, y finalmente se realizará un caso práctico en el cual se realiza efectivamente un modelamiento dinámico de un sistema.

2 | PROBLEMA DE INVESTIGACIÓN

Uno de los factores que permite tener un diseño y control eficiente de las operaciones de producción es la confiabilidad. “La evaluación de la confiabilidad/disponibilidad es un paso importante, frecuentemente indispensable, en el diseño y análisis de sistemas críticos” (Distefano y Puliafito, 2009). Por lo que para el cálculo de ésta métrica debe existir una precisión adecuada de los elementos considerados, tales como distribuciones estadísticas asociadas al comportamiento individual de los equipos, y distribución lógica del conjunto de máquinas, entre otros.

Para modelar la distribución lógica del conjunto de máquinas y analizar su confiabilidad existen herramientas estáticas y dinámicas. Dentro de las tradicionales o estáticas dos métodos ampliamente utilizados según Bourouni (2013) son reliability blocks diagrams (RBDs) y fault tree analysis (FT), éste último es ampliamente utilizado para identificar causas de fallas. Otros métodos tradicionales usados de acuerdo a Kim (2011) son reliability graphs (RGs), Cadenas de Markov, y simulación Monte Carlo.

En muchos artículos se señala que los métodos tradicionales no son suficientes para modelar ciertos comportamientos de sistemas complejos. En ese sentido Coit et al. (2015) señalan que el mayor límite de las técnicas tradicionales es la imposibilidad de modelar de cerca la naturaleza de los sistemas incluyendo variaciones dinámicas del medio ambiente y condiciones operacionales en que funciona el sistema. Desde otra perspectiva Barabadi et al. (2011) señalan que el hecho de que los modelos tradicionales son construidos en base al supuesto de independencia estocástica, es decir, que cada componente del sistema es estocásticamente independiente del resto, no se satisface, por ejemplo, cuando estos métodos son usados para modelar el comportamiento dinámico de un sistema causado por una variable dependiente del tiempo. También Distefano y Puliafito (2006) señalan al

respecto que los métodos comúnmente utilizados (RBD, FT y RG) tienen capacidad solo para modelar sistemas que no tienen relaciones entre las fallas de sus componentes.

Para solucionar las limitantes de las herramientas tradicionales existen las herramientas dinámicas, entre las que se cuentan dynamic reliability block diagram (DRBD), dynamic fault tree (DFT), y reliability graph with general gates. “El modelo DRBD extiende el tradicional modelo RBD considerando completamente varios tipos de dependencias y la dinámica del sistema” (Xu et al., 2008). Dynamic fault tree (DFT) y reliability graph with general gates también son extensiones de sus respectivos métodos tradicionales.

Mientras es bien conocido que los modelos FT y RBD son equivalentes, no es lo mismo para DFT y DRBD. El poder de dependencia del modelamiento de DRBD es mayor que DFT dado que DRBD puede modelar una dependencia “activa” relacionada con la reparación o activación de eventos, mientras DFT modela solo dependencias de “fallas” asociadas a eventos de fallo (Distefano et al., 2006).

Finalmente en relación al modelo DRBD Distefano et al. (2006) señalan que éste provee una aproximación más general para modelar la dependencia entre componentes o subsistemas que DFT. En DFT el concepto de dependencia es formalizado solo para representar redundancia, relaciones de orden o modos de falla común, mientras en DRBD es posible definir cualquier tipo de dependencia.

Dada la evolución al estudio de modelos dinámicos de confiabilidad presente en la literatura cabe preguntarse ¿Cómo efectivamente se pueden incorporar aspectos dinámicos al modelamiento de la confiabilidad del sistema?, ¿A qué tipo de sistemas se puede aplicar éste tipo de modelamiento?.

3 | OBJETIVOS

3.1. Objetivo General

Realizar una propuesta metodológica para aplicar el modelo DRBD mediante una herramienta de análisis.

3.2. Objetivos específicos

- Determinar los distintos elementos considerados en el modelamiento de la confiabilidad de sistemas.
- Determinar las propiedades básicas de un sistema dinámico de confiabilidad.
- Contextualizar sobre modelamiento dinámico de confiabilidad.
- Establecer los pasos a considerar para aplicar el modelo DRBD.
- Elaborar un modelamiento dinámico.

4 | MARCO TEÓRICO

4.1. Teoría clásica de confiabilidad

4.1.1. Confiabilidad

"La confiabilidad se define como la probabilidad que un elemento funcione sin fallar, durante un tiempo determinado bajo condiciones ambientales y de entorno preestablecidas" (Arata, 2009). "La confiabilidad es una medida que resume cuantitativamente el perfil de funcionalidad de un elemento y ayuda en el momento de seleccionar un equipo entre varias alternativas" (Melo et al., 2009).

Reconociendo que el tiempo al cual un elemento falla puede ser modelado como una variable aleatoria, y que además un elemento tiene dos estados posibles, en funcionamiento o en falla, la confiabilidad de éste es función solamente del tiempo, cuyas características dependen única y exclusivamente de la distribución de probabilidades con la que la falla pueda modelarse en el tiempo (Arata, 2009).

Dada una distribución de probabilidades se tiene una función de falla $f(t)$ o función de densidad probabilística de falla, que representa la probabilidad que un elemento falle en un instante de tiempo t cualquiera. Además existirá una probabilidad acumulada de falla $F(t)$, que cuantifica la probabilidad que el equipo (o sistema) falle en el intervalo $[0, t]$. Lo anterior significa que:

$$F(t) = \int_{t=0}^{t=t} f(t)dt \tag{4.1}$$

Mientras que la confiabilidad esta dada por:

$$R(t) = \int_{t=t}^{t=\infty} f(t)dt = 1 - \int_{t=0}^{t=t} f(t)dt = 1 - F(t) \tag{4.2}$$

"La confiabilidad de un elemento puede ser caracterizada a través de distintos modelos de probabilidades. Este modelamiento depende de la etapa de vida en que se encuentre el equipo" (Arata, 2009).

4.1.2. Tasa de falla

De acuerdo a los planteamientos de Arata (2009) el ciclo de vida de un equipo está dado por la tasa de falla. La que se define como "la probabilidad de falla en el intervalo de t a $t + dt$ asumiendo que el equipo sobrevive hasta el tiempo t" (Crespo, 2014).

De acuerdo a Høyland y Rausand (1994) la tasa de falla es frecuentemente alta en la fase inicial, lo que puede ser explicado por el hecho de que pueden haber efectos desconocidos en los ítems o componentes (conocidos como "mortalidad infantil"), los que pronto se muestran cuando los ítems son activados. Cuando el ítem ha sobrevivido la etapa de mortalidad infantil, la tasa de falla frecuentemente se estabiliza a un nivel por cierta cantidad de tiempo hasta que comienza a incrementarse tan pronto como los ítems comiencen a desgastarse. Esto tiene como resultado una curva con una forma característica denominada "curva de la bañera". Por lo cual, el ciclo de vida de un ítem puede ser dividido en 3 intervalos típicos: periodo de rodaje, periodo de vida útil, y periodo de desgaste. El comportamiento de la tasa de falla se puede ver representado en la Figura 4.1:

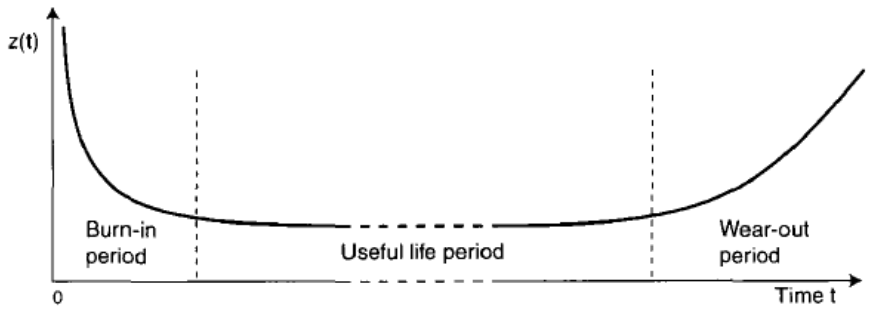


Figura 4.1: "Curva de la bañera"
(Fuente: Hoyland y Rausand, 1994)

4.1.3. MTBF, MTTF, MTTR

Conocido el comportamiento sobre la falla de un elemento, "es posible describir o identificar otro indicador importante de la seguridad de funcionamiento de un sistema. Este concepto es el tiempo medio entre fallas para equipos reparables o tiempo medio hasta la falla para los no reparables" (Arata, 2009). Estos conceptos son más conocidos como MTBF, (*Mean Time Between Failure*), y MTTF (*Mean Time To Failure*), respectivamente. Otro concepto asociado a las fallas de una componente es el tiempo promedio de intervención, que se define como: "es la esperanza matemática del tiempo antes de la puesta en servicio, o tiempo de no disponibilidad después de la falla" (Arata, 2009). Éste indicador se denomina MTTR (*Mean Time To Repair*).

4.1.4. Disponibilidad

Dados los conceptos definidos en los párrafos anteriores es posible definir el concepto de disponibilidad, que corresponde, de acuerdo a los planteamientos de Arata (2009), a la aptitud de un sistema de estar en un estado para cumplir una función requerida, en condiciones dadas, en el instante requerido y por un intervalo de tiempo requerido, suponiendo que está asegurada la provisión de los medios externos necesarios; es decir función correcta del equipo en el tiempo en que se requiera.

"La disponibilidad se define, matemáticamente, como la razón (o cuociente) que se establece entre el tiempo en que el sistema está, realmente disponible para el funcionamiento y el tiempo total, que incluye al tiempo anterior más el tiempo de reparación" (Arata, 2009).

Se sabe que "la disponibilidad está basada únicamente en la distribución de fallas y la distribución de tiempo de reparación. Esta puede además ser usada como un parámetro para el diseño" (Melo et al., 2009). Lo anterior se puede reflejar en la siguiente ecuación:

$$A = \frac{MTBF}{MTBF + MTTR} \tag{4.3}$$

Por lo tanto, y según lo señalado por Ebrahimi (2010) la disponibilidad se puede entender como la fracción del tiempo total que una componente, subsistema y/o sistema es utilizable.

Como se establece en los párrafos anteriores la disponibilidad puede ser aplicada a un sistema, sin embargo, es necesario conocer lo que significa un sistema en la teoría de confiabilidad.

4.1.5. Modelo de confiabilidad clásico

Un sistema es un conjunto de componentes, subsistemas y/o ensambles arreglados de acuerdo a un diseño específico con el fin de alcanzar un desempeño y niveles de confiabilidad aceptables. El tipo de componentes, sus cantidades, sus cualidades y la manera en que son arreglados dentro del sistema tienen un efecto directo en la confiabilidad del sistema. El mayor objetivo de la confiabilidad del sistema es la construcción de un modelo (distribución de vida) que represente el tiempo para fallar del sistema completo basado en las distribuciones de vida de los componentes, subsistemas y/o ensambles de los cuales se compone (Distefano y Puliafito, 2006).

"Existen muchos formalismos para modelar la confiabilidad de un sistema" (Distefano y Puliafito, 2006). De acuerdo a lo planteado por Estay (2014) un modelo clásico de confiabilidad corresponde al Diagrama de bloques de confiabilidad (RBD).

De acuerdo a Arata (2009) el modelo RBD representa gráficamente el funcionamiento del sistema mediante esquemas de bloques de confiabilidad, adecuadamente conectados entre sí, en los que cada bloque representa un subsistema o un componente. Las principales configuraciones RBD que detalla el autor son las siguientes:

Serie. Se define como *sistema en serie* cuando la falla de uno de sus elementos (cualquiera), que ha de considerarse como un acontecimiento independiente, determina la falla del sistema en un conjunto.

Esta configuración queda representada como se muestra en la Figura 4.2. Y la confiabilidad del sistema corresponderá a la probabilidad que todos los elementos (o subsistemas) no fallen en un tiempo determinado. Por lo tanto, la confiabilidad del subsistema corresponde al producto de las probabilidades de éxito de cada uno de los componentes:

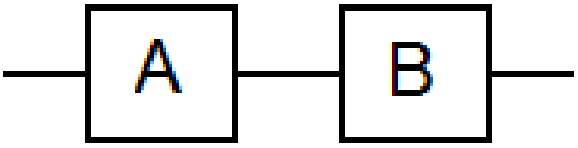


Figura 4.2: "Representación sistema en serie"
(Fuente: Elaboración propia)

$$R_S(t) = \prod_{i=1}^{n} R_i(t) \tag{4.4}$$

Donde $R_S(t)$ corresponde a la confiabilidad del sistema, $R_i(t)$ es la confiabilidad de cada componente o subsistema en serie, y n es el número de componentes.

Redundancia total (paralelo). Se da cuando en el sistema un elemento, por sí solo, es capaz de soportar el buen funcionamiento del sistema.

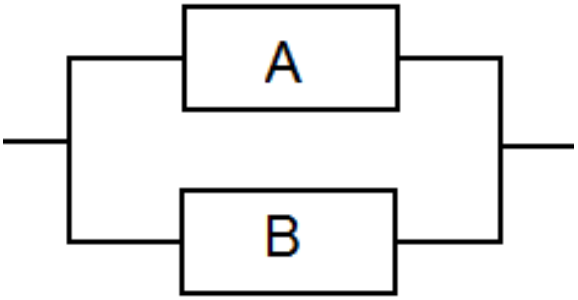


Figura 4.3: "Representación sistema en paralelo"
(Fuente: Elaboración propia)

La Figura 4.3 muestra esta configuración con 2 elementos.

Se considera que la falla del sistema ocurre cuando fallan simultáneamente *todos* los elementos que lo componen. Por lo tanto se tiene que la confiabilidad del sistema puede ser representada de acuerdo a la siguiente ecuación:

$$R_S(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)) \tag{4.5}$$

Redundancia parcial. Es la configuración en la cual un *grupo* de elementos es capaz de soportar el buen funcionamiento del sistema.

Se representa del mismo modo que la redundancia total, sin embargo, como una cierta combinación mínima de los elementos debe estar en operación para que el sistema funcione, la confiabilidad del sistema estará dada por:

$$R_S(t) = \sum_{j=r}^{n} \binom{n}{j} R^j (1-R)^{n-j} \tag{4.6}$$

Donde $R_S(t)$ es la confiabilidad para un determinado tiempo t en un sistema compuesto por n elementos iguales, de los cuales se requiere un mínimo de r equipos en un buen funcionamiento para que el sistema funcione. Por otra parte R corresponde a la confiabilidad de cada componente de manera individual.

Stand-by. Consiste en que en un instante determinado funciona sólo en uno de los elementos del sistema, mientras que los restantes permanecen en reserva, en estado de espera.

Un sistema en stand-by queda representado de la siguiente forma:

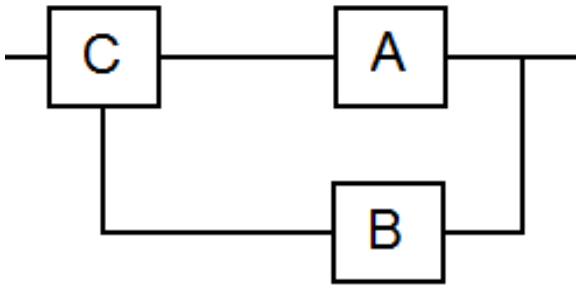


Figura 4.4: "Representación sistema en Stand-by"
(Fuente: Elaboración propia)

En la Figura 4.4 se tienen las componentes A y B y un conmutador, C.

La configuración stand-by además considera dos escenarios:

- **Stand-by en frío.** En esta el equipo de reserva no se encuentra en operación, por lo que no existe pérdida de confiabilidad en él. Sin embargo, existe un mayor tiempo de set-up para su puesta en operación.

- **Stand-by en caliente** El equipo de reserva si está en operación, existiendo pérdida de confiabilidad pero menor tiempo de set-up.

Si se considera una tasa de falla constante e igual para los equipos, la confiabilidad de un sistema en *stand-by* en frío se calcula de acuerdo a:

$$R_S(t) = e^{-\lambda t}(1 + \lambda t)R_C(t) \tag{4.7}$$

Donde $R_S(t)$ y $R_C(t)$ corresponden a la confiabilidad del sistema y a la confiabilidad del conmutador respectivamente.

Fraccionamiento. n equipos se reparten de forma proporcional o no la carga del trabajo. Pueden tener capacidad ociosa: se puede operar una fracción de la carga total.

4.2. Modelo DRBD

De acuerdo a lo señalado por Xu et al. (2008) el modelo RBD antes expuesto encuentra su extensión en el modelo DRBD.

La herramienta dinámica de modelamiento de sistemas DRBD puede ser entendida esencialmente de acuerdo a lo señalado por Distefano et al. (2006) y Distefano y Puliafito (2006), en donde se establece en primer lugar dos aspectos dinámicos que integra el modelo:

- La variación con el tiempo, que es integrada como una característica del sistema si el estado de una componente evoluciona cuando una secuencia de eventos ocurre.
- La dependencia entre componentes, integrada a la confiabilidad de las relaciones entre componentes por medio de la asociación de eventos a tales relaciones.

4.2.1. El modelo dinámico

En el modelo DRBD la condición de cada componente es caracterizada por un estado variable identificando la condición operacional de la componente en un tiempo dado. La evolución del estado de una componente (dinámica de la componente) es caracterizada por los **eventos** que le ocurren a ésta. El evento de una componente es definido como un cambiante de estado. Esta operación es considerada instantánea y atómica en el dominio DRBD (Distefano et al., 2006).

La dinámica del sistema en el tiempo t es definida por los estados alcanzados por sus componentes en el instante t y la secuencia de eventos ocurridos hasta t (Distefano et al., 2006).

La máquina de estado finito caracterizada en la Figura 4.5 representa todos los **estados** (rectangulos redondeados) y **eventos** (flechas) que una componente DRBD puede asumir o realizar.

Los estados son:

- **Activa** - Una componente esta en este estado cuando funciona sin ningún problema.

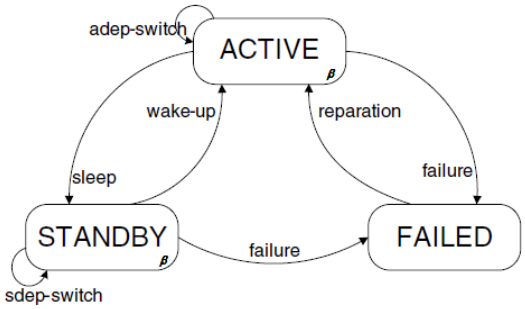


Figura 4.5: "Máquina estados-eventos DRBD"
(Fuente: Distefano, Scarpa, y Puliafito, 2006)

- **Fallada** - Una componente en este estado no es operacional, seguida de su falla. Para restaurar una componente fallada debe ser *reparada*.
- **Stand-by** - Una componente en este estado no esta habilitada para trabajar. Sin embargo, el estado staand-by es reversible: una componente en stand-by puede ser reactivada sin una reparación específica. Un estado stand-by es siempre consecuencia de la aplicación de una *dependencia* a la componente correspondiente. Es importante remarcar que, desde el punto de vista de la confiabilidad, una componente en stand-by es considerada inactiva o no operacional como una componente fallada por el intervalo de tiempo completo en que persiste en este estado (Distefano y Puliafito, 2006).

Un evento representa la transición del estado de una componente a otro. Los eventos posibles, como lo reporta la Figura 4.5, son:

- **Failure** - Evento que representa la transición desde un estado activo o stand-by a uno fallado.
- **Wake-up** - Evento que representa la transición de un estado stand-by a uno activo.
- **Sleep** - Evento que representa la transición de un estado activo a uno stand-by.
- **Reparation** - Evento que representa la reparación de una componente fallada a su consecuente reactivación.

- **Adep-switch** - Transición entre dos estados activos debido al cambio de la dependencia aplicada.

- **Sdep-switch** - Transición entre dos estados stand-by en correspondencia con el cambio de la dependencia aplicada (Distefano y Puliafito, 2006).

"Los últimos eventos estan relacionados con la propiedad de *concurrencia* de las dependencias: si dos dependencias son satisfechas simultáneamente, un switch entre el estado activo o stand-by respectivo es necesario, de acuerdo al tipo de dependencia"(Distefano et al., 2006).

Reparation y failure son eventos externos, no dependientes de ningún otro evento DRBD, como lo pueden ser wake-up, sleep, adep-switch y sdep-switch. En particular la reparación requiere de intervención externa modelada por una política de *mantenimiento* (preventiva o correctiva) probabilísticamente representada por su función de distribución acumulada (Distefano y Puliafito, 2006).

De acuerdo a lo planteado por Distefano y Puliafito (2006), cada componente DRBD está caracterizada también por una cantidad probabilistica analítica representando su confiabilidad. Esta cantidad es la tasa de falla dada por la función $z(t)$, la que una vez definida permite alimentar al modelo DRBD con información real.

4.2.2. Relaciones de dependencia

Una dependencia establece una relación de confiabilidad entre dos componentes o subsistemas, un *driver* y un *target*. Cuando un evento específico, llamado *action* o *trigger*, ocurre al driver, la condición de dependencia es aplicada al target. Esta condición es asociada a un evento específico del target, llamado *reaction*, y será verificado y por lo tanto aplicado hasta que el *driver* permanezca en el estado alcanzado dada la ocurrencia del evento *trigger*. Cuando esta condición se vuelva insatisfecha, la componente target será puesta en estado activo (Distefano et al., 2006).

La dependencia es la herramienta para modelar varios aspectos de confiabilidad dinámica de sistemas genéricos en el dominio de DRBD, tales como interdependencia entre componentes o subsistemas, efectos debidos a load sharing, interferencia, redundancia

standby, causas de falla común, y también representar correctamente aspectos de reparabilidad (Distefano et al., 2006).

Una dependencia es representada usando una línea punteada con un círculo en la terminación de la componente target que representa la conexión entre componentes relacionadas como se muestra en la Figura 4.6 (Distefano et al., 2006):

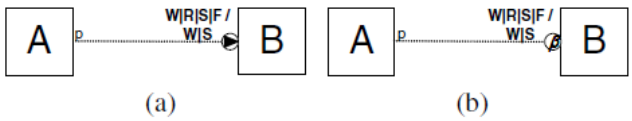


Figura 4.6: "Representación de dependencias (a) de orden y (b) fuerte en DRBD"
(Fuente: Distefano, Scarpa, y Puliafito, 2006)

Las dependencias como lo señala Distefano y Puliafito (2006) pueden ser clasificadas de acuerdo a la acción, la reacción, el tipo de relación y otras particularidades:

Acción. Cuatro tipos de eventos pueden ser identificados, correspondiendo con los siguientes tipos de dependencia:

Wake-up (W) - La dependencia es aplicada en caso de un evento trigger *wake-up*.

Reparation (R) - La dependencia es aplicada en caso de un evento trigger *reparation*.

Sleep (S) - La dependencia es aplicada en caso de un evento trigger *sleep*.

Failure (F) - La dependencia es aplicada en caso de un evento trigger *failure*.

Reacción. Dos grupos de dependencias son distinguidos considerando el evento reacción: **Wake-up (W)** y **Sleep (S)**.

Relación. Considerando que la dependencia puede modelar la relación entre dos componentes, dos principales clases de dependencias son individualizadas:

Orden - Establece el orden secuencial entre un evento trigger y un evento reaction: condición satisfecha si el trigger ocurre siempre antes o simultáneamente con la reacción.

Fuerte - Fuerza a la componente target a reaccionar cuando el evento trigger ocurre. En caso de dependencia fuerte la componente target es activada (reacción wake-up W) o desactivada (reacción Sleep S) cuando el evento trigger le ocurre al driver de acuerdo a la *tasa de dependencia* β.

Las clasificaciones de dependencias en los eventos reaction, necesitan diferenciarse dependiendo de la relación de dependencia. En el caso de la dependencia de orden los cuatro eventos caracterizando el trigger podrían también ser considerados para el evento reaction: *wake-up (W)*, *reparation (R)*, *sleep (S)* y *failure (F)*. Para la dependencia fuerte solo dos eventos reaction son definidos considerando dos posibles estados de llegada, standby y active: *wake-up (W)* y *sleep (S)* (Distefano et al., 2006).

En el ejemplo de la Figura 4.6, A es una componente driver y B es la componente target. La acción de dependencia o evento trigger se encuentra indicado por una letra (W, R, S o F), y el evento reaction por otra letra (W o S) separada del evento trigger por una barra. La cadena completa se posiciona cerca del círculo. En caso de dependencias de orden una flecha es posicionada dentro del círculo, mientras un número caracteriza las dependencias fuertes: indica la tasa de dependencia asociada a una dependencia. Cuando es omitida se considera un valor por defecto de $\beta = 1$ (Distefano et al., 2006).

La tasa de dependencia β caracteriza la dependencia fuerte, pondera el impacto, en términos de confiabilidad, al target debido a la ocurrencia de un evento trigger en la componente driver. Si z_A y z_B son las tasas de falla independientes de las componentes representadas en la Figura 4.6 (b), y el β es la tasa de dependencia, si se mantiene una relación de dependencia fuerte, la tasa de falla de la componente target (B) se convierte en:

$$z_B^{Dep}(t) = \beta z_B(t) \tag{4.8}$$

De acuerdo a una reacción de una dependencia fuerte, existen varias posibles interpretaciones del parámetro β. En el caso de W-reaction, como consecuencia de la ocurrencia de un trigger, el target será cambiado a un estado activo donde β representa la actividad de la componente. Esto podría modelar load sharing, modo de falla común o probabilístico, y muchos otros comportamientos de confiabilidad. En caso de S-reaction, la ocurrencia

del trigger cambiará al target a un estado stand-by, energizado proporcionalmente a β (Distefano y Puliafito, 2006).

Considerando los valores que el parámetro β podría asumir, cuatro tipos de estados *activo* y/o *stand-by* podrían ser caracterizados (Distefano et al., 2006):

- *Idealmente activo y/o cold stand-by* $\beta = 0$*:* una componente en éste estado no puede fallar: Una componente idealmente activa esta operacional siempre con tasa de falla cero; una componente en cold standby está deshabilitada y a la vez no energizada.

- *Parcialmente activo y/o warm stand-by* $0 < \beta < 1$*:* Los valores de β en éste rango modelan condiciones en que la componente dependiente (target) incrementa su confiabilidad si la condición de dependencia es satisfecha. Una componente en estado warm standby esta deshabilitada, pero parcialmente energizada, mientras en el estado parcialmente activo está operacional con una confiabilidad más alta que un estado completamente activo (de no dependencia).

- *Completamente activo y/o hot stand-by* $\beta = 1$*:* el target se encuentra totalmente activado o totalmente energizado pero deshabilitado (hot standby) luego de la aplicación de la dependencia. Una componente que no está bajo el efecto de alguna dependencia podría estar fallada o completamente activa.

- *Hiperactivo y/o burning stand-by* $\beta > 1$*:* La componente dependiente desciende su confiabilidad cuando la condición de dependencia es satisfecha. Una componente hiperactiva esta en actividad alterada, mientras una componente deshabilitada pero sobre energizada está en burning standby.

En el formalismo DRBD las dependencias también pueden ser compuestas. La composición permite definir dos o más dependencias entrantes a la misma componente target. Por otra parte, la composición introduce el problema de la *concurrencia entre las dependencias*: dos o más dependencias podrían ser verificadas simultáneamente. Para resolver éste problema un mecanismo de prioridad es explotado: Un modelador podría discriminar las dependencias por medio de la asociación de una ponderación numérica a cada una de ellas. Este valor, llamado *prioridad de conexión*, es posicionado cerca de la componente

driver (p en la Figura 4.6), y podría también ser omitido. En éste caso la prioridad es dada a las dependencias con un valor especificado. En caso de conflicto entre dependencias con la misma (o sin especificar) prioridad de conexión, la evaluación está hecha considerando la acción, la reacción y la relación de las dependencias involucradas identificando una *jerarquía de prioridad* (Distefano et al., 2006).

4.2.3. Modelamiento de redundancia

"En el formalismo DRBD, la redundancia puede ser fácilmente representada por medio de la explotación y combinación de dependencias entre unidades. Una componente redundante es generalmente representada como una conexión paralela de sus unidades o instancias simples" (Distefano y Puliafito, 2006).

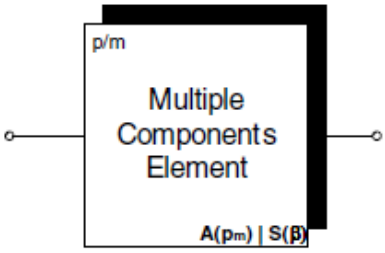


Figura 4.7: "Elemento de componente múltiple DRBD"
(Fuente: Distefano, Scarpa, y Puliafito, 2006)

A modo de hacer el modelamiento DRBD más fácil de usar y más compacto Distefano y Puliafito (2006) introduce un elemento de sintaxis de *componente múltiple* como lo muestra la Figura 4.7.

Una componente múltiple es una alternativa compacta para representar algunas implementaciones específicas de múltiples unidades de componentes redundantes. Está compuesta por dos o más *m instancias* de componentes básicas en una estructura paralela de conexión. Todas las instancias realizan las mismas tareas específicas. La *multiplicidad m* de las componentes es situada en la esquina superior izquierda, mientras la esquina inferior derecha especifíca la política de redundancia aplicada y el valor β (Distefano y Puliafito, 2006).

Una componente múltiple puede ser conectada a otras componentes en estructuras de serie o paralelo y también es posible especificar toda clase de dependencias entrantes o salientes desde/hacia ella. Las dependencias entrantes son heredadas por todas las instancias activas de las componentes básicas que reaccionarán apropiadamente a todos los eventos trigger ocurriendole a los drivers (Distefano y Puliafito, 2006).

La *política de redundancia* caracteriza cómo las unidades redundantes son manejadas. Dos alternativas diferentes son posibles, política de redundancia *activo* o *stand-by*, representada por la letra mayúscula en la esquina inferior derecha. En una componente múltiple manejada por una *política de redundancia activa* todas las unidades redundantes estan operacionales (activas) al mismo tiempo. También implementa una componente redundante k de cada m (k out of m): si al menos k de m instancias estan operacionales la componente es operacional. El número entre paréntesis de símbolo $A(p_m)$ indica una política de redundancia activa, p_m, representa la probabilidad de modo de falla común. En caso de *política de redundancia stand-by*,algunas instancias (k) estan simultaneamente activas, y las otras ($m-k$) estan en estado stand-by, habilitadas en caso de falla de acuerdo a un orden especificado. El estado stand-by de las instancias deshabilitadas es caracterizado por el parámetro de dependencia β presentado en la esquina inferior derecha ($S(\beta)$). k y m son en ambos casos presentados en la esquina superior izquierda de la representación de la componente múltiple de la Figura 4.7 (Distefano et al., 2006).

Según lo establecido por (Distefano y Puliafito, 2006) se pueden modelar dos tipos de política de redundancia stand-by:

- **Redundancia cold stand-by.** ($\beta = 0$). Caso en que la instancia dependiente es activada si y solo si las instancias previas fallan.
- **Redundancia partly loaded stand-by.** ($0 < \beta \leq 1$). Caso en que cada componente depende directamente de todas las componentes previas, dado que la componente puede fallar antes que las unidades redundantes activas fallen.

La Figura 4.8 es una representación de las políticas de redundancia stand-by señaladas anteriormente.

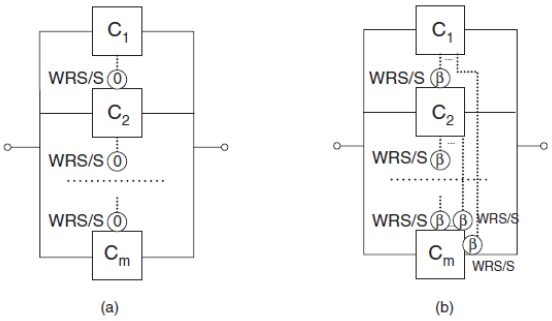


Figura 4.8: "Modelos DRBD de redundancia stand-by cold (a) y partly loaded (b)"
(Fuente: Distefano y Puliafito, 2006)

4.2.4. Modelamiento de load sharing

"Load sharing es una condición en la cual dos o más componentes comparten la misma carga de trabajo. Por lo tanto, la condición de load sharing caracteriza componentes que desempeñan tareas similares"(Distefano y Puliafito, 2006).

Para modelar cambios en la confiabilidad de acuerdo a variaciones de la carga compartida entre dos o más componentes, el concepto de dependencias es explotado. Este comportamiento podría ser modelado mejor mediante una dependencia de W-reaction, cuantificando el impacto a través de la tasa de dependencia β (Distefano y Puliafito, 2006).

En la Figura 4.9 se representan dos capacitores (a), se sabe que ambos deben fallar en corto circuito para que el sistema falle. Sin embargo, de acuerdo a los planteamientos de Distefano y Puliafito (2006) la conexión entre ellos no es simplemente paralelo porque si un capacitor falla (corto circuito) entonces el voltaje aplicado al restante será duplicado y su tasa de falla aumentará considerablemente. La situación puede ser modelada por medio de una estructura en paralelo donde las dos componentes son mutuamente dependientes, como lo presentado por la Figura 4.9 (b): inicialmente la carga es compartida entre las dos componentes como efecto de la aplicación de las dependencias recíprocas; cuando una de ellas falla, toda la carga va hacia la otra que será completamente activada.

Como la muestra la Figura 4.9 (b) y según Distefano y Puliafito (2006) el comportamiento load sharing se modela mediante dependencias fuertes W/W (wake-up/wake-up) caracterizadas por $(0 < \beta_1, \beta_2 < 1)$. En ellas cuando ambas componentes se vuelven activas las dependencias son simultáneamente aplicadas. La dependencia cesa si una componente

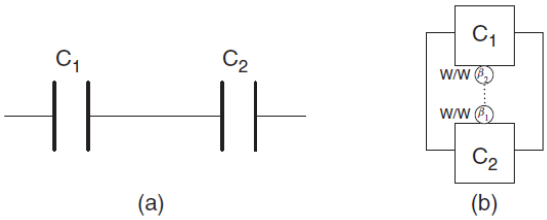


Figura 4.9: "Modelo DRBD de load sharing"
(Fuente: Distefano y Puliafito, 2006)

falla, además la otra componente vuelve a un estado completamente activo.

El modelamiento realizado en Distefano y Puliafito (2006) cuenta con que la tasa se dependencia sea calculada considerando la tasa de servicio y la carga asignada a cada componente involucrada.

4.2.5. Ejemplo de modelamiento de sistemas

En Distefano y Puliafito (2009) se pueden encontrar el modelamiento de un sistema computacional distribuido multiprocesador.

El sistema se compone de dos módulos computacionales: CM_1 y CM_2. Cada uno de ellos cntiene un procesador (P_1 y P_2, respectivamente), una memoria (M_1 y M_2) y dos discos duros: uno primario (D_{11} y D_{21}) y un disco de respaldo (D_{12} y D_{22}). Inicialmente el disco primario es accedido mediante el módulo computacional correspondiente, mientras el disco duro de respaldo contiene la copia de la información del disco duro primario, y es accedido solo periodicamente para operaciones de actualización. Si el disco primario falla, es reemplazado en su función por el disco de respaldo. En términos de confiabilidad los discos son idénticos, son caracterizados por la misma tasa de falla o función de distribución acumulada de confiabilidad (cdf). Pero, cuando el disco duro primario está operacional, la tasa de falla del disco duro de respaldo decrece como consecuencia de las condiciones de menor carga. (Distefano y Puliafito, 2009)

Los módulos computacionales se encuentran conectados por medio del bus N; además, P_1 y P_2 se encuentran energizados por la fuente de poder PS: la falla de PS fuerza P_1 y P_2 a fallar.

M_3 es una memoria de repuesto que reemplaza M_1 y M_2 en el caso defalla. Si ambas M_1

y M_2 se encuentran operativas, M_3 se mantiene solo viva o bajamente energizada con el fin de mantener la información almacenada, pero no es accedida para leer/escribir información. Cuando M_1 o M_2 o ambas fallan, M_3 substituye la unidad fallada.

Con el fin de trabajar adecuadamente el sistema computacional multiprocesador requiere que al menos un módulo computacional (CM_1 o CM_2), la fuente de poder PS, y el bus N esten operando correctamente. Además, un módulo computacional (CM_1 y CM_2) es operativo si un procesador (P_1 y P_2, respectivamente), una entre las memorias locales (M_1 y M_2) y la memoria compartida M_3, y un disco duro (D_{11} o D_{12} para CM_1 y D_{21} o D_{22} para CM_2) no se encuentre fallado.

El ejemplo descrito en los párrafos precedentes queda representado de la siguiente forma:

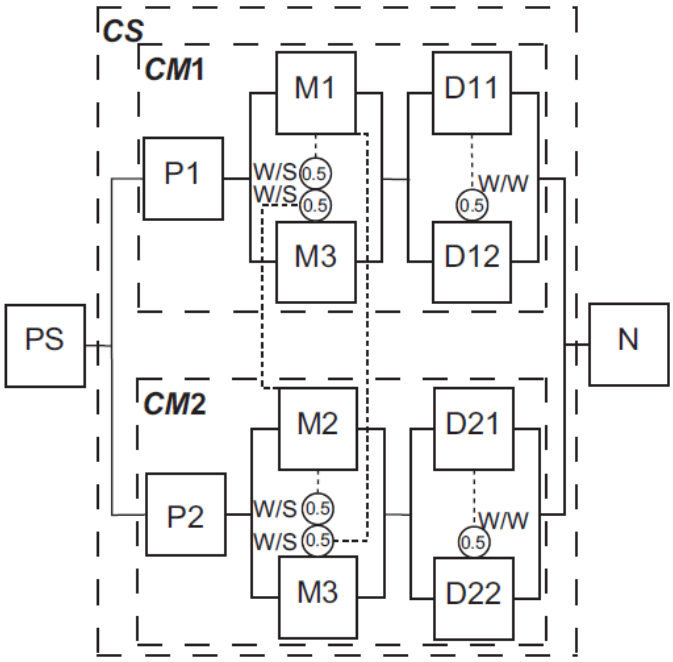


Figura 4.10: "Modelo DRBD Sistema computacional distribuido Multiprocesador"
(Fuente: Distefano y Puliafito, 2009)

En el modelo DRBD de la Figura 4.10 cada dispositivo del sistema es representado por la correspondiente componente caracterizada por su correspondiente función de distribución acumulada de confiabilidad. El comportamiento de energizado de la fuente de poder PS a los procesadores P_1 y P_2 se modela como una simple serie entre cada procesador y la fuente de poder.

La política de manejo de los discos duros es representada como una dependencia fuerte Wake-up/Wake-up: cuando el disco primario D_{11} y/o D_{21} se encuentra operacional, los discos de respaldo D_{12} y/o D_{22}, respectivamente se encuentran parcialmente activos manteniendo la copia de seguridad. La tasa de dependencia elegida en éste caso es $\beta = 0,5$. Cuando el disco primario falla el disco substituto se vuelve completamente activo y completamente energizado dado que ya no se satisface la condición de dependencia.

Para modelar la política de redundancia de las memorias se utiliza la dependencia Wake-up/Sleep (W/S). Esta permite representar las condiciones deshabilitantes aplicadas por M_1 y M_2, cuando se encuentran operacionales, sobre M_3, manteniendo ésta última en stand by. La dependencia completa debe ser aplicada a M_3 si y solo si ambas M_1 y M_2se encuentran simultaneamente operacionales; cuando, al menos, una de ellas falla, M_3 debe cambiar al estado completamente activo. Para implementar esta condición las dos dependencias W/S, desde M_1 a M_3 y desde M_2 a M_3 son series compuestas: cuando ambas son simultaneamente satisfechas, la componente M_3 se encontrará en stand by, de lo contrario M_3 estará activa. Esta dependencia compuesta es duplicada en el modelo DRBD por motivo de claridad, identificando y separando los dos módulos computacionales CM_1 y CM_2.

4.2.6. Especificación formal del modelo

En Distefano y Puliafito (2009) se establece la formalización del modelo DRBD, la que se explica en los siguientes párrafos.

Un modelamiento DRBD posee un grafo conectado bipartito cuyos nodos son unidades ($\mathbb{U}$), y sus arcos, conexiones de nodos en $\mathbb{U}$. Se tienen 2 tipos de conexiones permitidas: *independiente* ($\mathbb{I}$) y *dependiente* ($\mathbb{D}$).

Formalmente, un modelo DRBD $\mathscr{M}$ puede ser descrito analíticamente por:

$\mathscr{M} \stackrel{\text{DEF}}{=} (\mathbb{U}, \mathbb{I}, \mathbb{D})$

donde:

- $\mathbb{U}$ es el conjunto de las unidades;
- $\mathbb{I}$ es el conjunto de conexiones independientes y

- $\mathbb{D}$ es el conjunto de conexiones dependientes o dependencias.

Una unidad DRBD genérica $u \in \mathbb{U}$ es descrita por el conjunto de pares de sus estados $\mathbb{S}_u$.

Un estado $s_u \in \mathbb{S}_u$ de u es especificado por el par:
$s_u \overset{\text{DEF}}{=} (T, F(\cdot))$
donde:

- $T \in [Activo, Standby, Fallado]$ es el tipo de estado y
- $F : \mathbb{R}^+ \to [0, 1] \subset \mathbb{R}$ es la función de distribución acumulada de confiabilidad (cdf) de la unidad u en el estado s si $T \neq Fallado$; es la función de distribución acumulada de mantenimiento de u si $T = Fallado$.

El estado completamente activo de una unidad u es $s_u^0 \overset{\text{DEF}}{=} (Activo, F^0(\cdot))$ donde $F^0(\cdot)$ es la función de distribución acumulada de confiabilidad inicial de u. El estado del sistema en el tiempo t, $S(t)$, se define como el estado alcanzado por todas sus n unidades en t, $s_i(t)$ con $i = 1, ..., n$, o $S(t) \overset{\text{DEF}}{=} (s_1(t), ..., s_n(t))$.

Un evento DRBD $e_u \in \mathbb{B}_u$ representa un cambio de estado de una unidad u. Es expresado como el par de los estados inicial y final $e_u \overset{\text{DEF}}{=} \{s_u^i, s_u^j\}$, donde $s_u^i, s_u^j \in \mathbb{S}_u$ y $i, j = 1, ..., m_u$; $i \neq j$.

$\mathbb{B}_u$ es el conjunto de todos los eventos que pueden ocurrirle a la unidad u. El conjunto de todos los eventos del modelo DRBD $\mathscr{M}$ es identificado como $\mathbb{B} = \bigcup_{u \in \mathbb{U}} \mathbb{B}_u$.

Cada evento que le ocurre a una unidad DRBD es caracterizado por la variable aleatoria tiempo ocurrido hasta el evento. El conjunto de todas las variables aleatorias asociadas a una unidad DRBD constituye el proceso estocástico de tiempo continuo $\mathfrak{B} = \{P_t \in \Omega; t \geqslant 0\}$ que toma valores en el conjunto numerable de espacio de estados discreto Ω compuesto por todos los estados que la unidad DRBD puede asumir. $\mathfrak{B}$ representa la dinámica de las unidades.

Una *conexión independiente* $c \in \mathbb{I}$ representa un arco entre dos unidades, y por lo tanto es identificado por el par de unidades $c = (a \in \mathbb{U}, b \in \mathbb{U})$. La confiabilidad del sistema es obtenida aplicando la teoría de probabilidad (AND, OR, teorema de probabilidad total) a

las funciones de distribución acumulada de las unidades, o las ecuaciones de estructuras analíticas.

Formalmente, una conexión de *dependencia simple* o *dependencia* $d \in \mathbb{D}$ se define como la 4-tupla:

$$d \overset{\text{DEF}}{=} (tr, re_{Tg}, p, c) \tag{4.9}$$

donde:

- $tr \in \mathbb{B}$ es el evento trigger;
- $re_{Tg} \in \mathbb{B}$ es el evento reaction de la unidad/subsistema target $Tg \in \mathbb{C}$;
- $p \in [0, 1] \subset \mathbb{R}$ es la *probabilidad de propagación* y
- $c \in \mathbb{N}$ es la *prioridad de conexión*.

Bajo ésta nomenclatura la *dependencia genérica* del modelo DRBD descrita anteriormente queda de la siguiente forma:

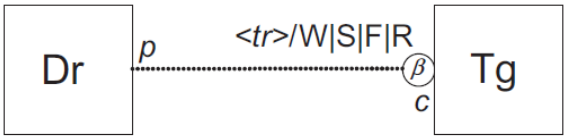


Figura 4.11: "Dependencia genérica DRBD"
(Fuente: Distefano y Puliafito, 2009)

Inicialmente en el instane de tiempo $t = 0^-$, las unidades componiendo el modelo DRBD $\mathscr{M}$ estan considerados en un estado stand-by en frío $s_{u_1}(0^-), ..., s_{u_n}(0^-)$.

En $t = 0$ los eventos $\{Wakeup^0_{u_1} = (s_{u_1}(0^-), s^0_{u_1}), ..., Wakeup^0_{u_n} = (s_{u_n}(0^-), s^0_{u_n})\} \in \mathbb{B}$ se ven obligados por todos ellos, el evento *big bang*, dejando a todas las unidades en un estado completamente activo $s^0_{C_1}$ y desencadenando las reacciones de las dependencias involucradas en las respectivas unidades target. Las unidades involucradas en las reacciones de dependencia *big bang* son establecidas en t^+_0, mientras las otras se encuentran completamente activas. En consecuencia en t^+_0 el estado inicial $S(0) = (S_{C_1}(0), ..., S_{C_n}(0))$ es inequivocamente identificado.

La composición es una operación basicamente aplicada a dos dependencias entrantes a la misma unidad target $u \in \mathbb{U}$, representada como la función:

$$y_u : \mathbb{D} \times \mathbb{D} \times \mathbb{R}^+ \rightarrow \mathbb{D} \tag{4.10}$$

Función que, fijado el instante de tiempo t, especifica la dependencia aplicada a u. Ya que, el resultado de la composición de dependencias es también una dependencia, aplicando recursivamente la composición, es posible extender tal operación a más de 2 dependencias.

4.3. FT y DFT

Un acercamiento al modelo FT y DFT puede ser encontrado en Distefano y Puliafito (2007), y será expuesto en los siguientes párrafos.

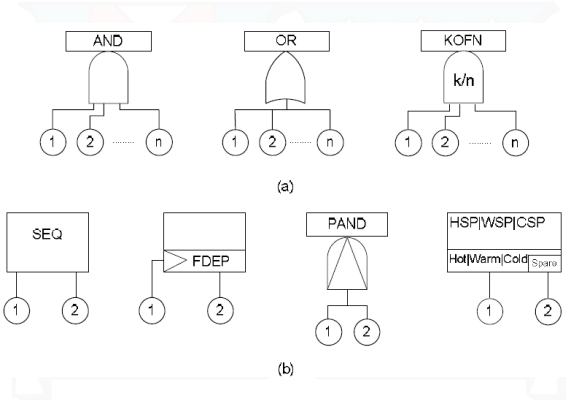


Figura 4.12: "Elementos de FT (a) y DFT (b)"
(Fuente: Distefano y Puliafito, 2007)

El modelo Fault Trees ha sido ampliamente usado en el análisis de la confiabilidad por sobre cuarenta años. En este se explota compuertas booleanas para representar cómo las fallas de las componentes se combinan para producir la falla del sistema. Dynamic Fault Trees agrega una noción temporal, desde que las fallas del sistema pueden depender del orden de falla de las componentes. Las compuertas estáticas son por lo tanto puramente compuertas lógicas combinatorias aplicadas a los eventos de falla de entrada. Estas son representadas en la Figura 4.12 (a) e informalmente definidos como sigue:

- **AND** - El evento de falla saliente ocurre si ocurren todos los eventos de entrada no repetidos.
- **OR** - Falla si cualquier entrada falla.
- **KOFN** - Falla si k de n entradas no repetidas fallan.

Aunque fault trees estático es comunmente usado, es limitado en el modelamiento de sistemas que no tienen relaciones secuenciales entre las fallas de sus componentes. Las secuencias de falla de las componentes son mejor capturadas usando Dynamic fault trees.

Dynamic fault trees usa compuertas como los elementos dinámicos primarios de la Figura 4.12 (b):

- ***Compuerta Sequence Enforcing (SEQ)*** - Las compuertas SEQ afirman que la falla de la componente solo puede ocurrir en un orden de secuencia especificado por sus entradas;

- ***Compuerta Functional Dependency (FDEP)*** - La compuerta FDEP propaga la falla gatillante de entrada a los eventos básicos dependientes;

- ***Compuerta Priority AND (PAND)*** - Las compuertas PAND fallan si las entradas no repetidas fallan en un orden dado;

- ***Puertos Cold, Hot y Warm Spare (CSP, HSP y WSP*** - Las compuertas CSP, HSP y WSP modelan relaciones de unidades de repuesto. Cuando la entrada primaria falla, las entradas de repuesto disponibles son usadas en orden hasta que no queda ninguna, en cuyo tiempo la compuerta falla. La *"temperatura´´* de la compuerta (factor de inactividad) selecciona el grado en que la tasa de falla del repuesto es atenuada.

Dynamic fault trees utiliza modelos de Markov como representaciones analíticas. Estos modelos de Markov (CTMC) son convertidos en ecuaciones diferenciales y resueltos numericamente. Alternativas válidas a CTCM en el análisis de DFT se representan por Petri nets, Bayesian networks, y simulación.

4.4. DRBD vs DFT

Los mismos autores en Distefano y Puliafito (2007) dan a conocer las diferencias entre DRBD y DFT, las que se resumen en lo siguiente:

Enfoque del modelamiento: Mientras FT y DFT representan solo las fallas de las componentes, RBD y DRBD pueden además modelar reparaciones y activaciones.

Enfoque del modelamiento de comportamientos dinámicos de confiabilidad: DFT define compuesrtas específicas para representar dependencia de falla y/o modo de falla común (FDEP), redundancia (Compuertas Spare) y relaciones de orden (Compuertas SEQ y PAND), mientras DRBD explicíta y formaliza el concepto de dependencia entre componentes como la base del modelamiento de cualquier aspecto dinámico.

Estas diferencias muestran que el modelo DRBD por una parte puede modelar mayor cantidad de aspectos dinámicos, y por otra ofrece mayor grado de libertad en cuanto a representación. También queda establecido que no se mantiene la equivalencia entre el modelo DRBD y DFT, como si existe entre RBD y FT.

Es importante mencionar que en FT (DFT) un sistema es caracterizado por las fallas de sus componentes, mientras en RBD (DRBD) el sistema queda representado directamente por sus componentes.

4.5. El modelo DRBD bajo otra línea de investigación

Uno de los desarrollos más tempranos sobre el modelo DRBD se puede encontrar en Distefano y Xing (2006), de éste trabajo se rescata la simplicidad de lo que se modela como "dependencia" (entre otras características) en los trabajos posteriores de Distefano en colaboración con otros autores. Esta se plantea en un principio como lo que se puede apreciar en la Figura 4.13, lo cual no dista mucho del modelo más reciente.

Figura 4.13: "Representación de dependencia genérica DRBD"
(Fuente: Distefano y Xing, 2006)

Otra línea de investigación se habre a partir de Distefano y Xing (2006), donde Xing trabaja en colaboración con otros autores, como por ejemplo Liudong y Xu (2007) y Xu et al. (2008) donde se establece que las creaciones DRBD introducidas en Distefano y Xing (2006) son muy complicadas y difíciles de usar, por lo que no serían practicamente usables. Lo señalado anteriormente por los autores se puede ver reflejado en los símbolos y acrónimos para dependencias DRBD que fueron creados en dicho desarrollo temprano, Figura 4.14:

	Activation								Failure							
	Direct		Inverse						Direct				Inverse			
	Simple	Strict	Simple			Strict			Simple			Strict			Simple	Strict
Strong	ADSD	SADSD	Cold CAISD	Warm WAISD	Hot HAISD	Cold SCAISD	Warm SWAISD	Hot SHAISD	Cold CFDSD	Warm WFDSD	Hot HFDSD	Cold SCFDSD	Warm SWFDSD	Hot SHFDSD	FISD	SFISD
	A	A	A	A	A	A	A	A	F	F	F	F	F	F	F	F
Order	ADOD	SADOD	AIOD			SAIOD			FDOD			SFDOD			FIOD	SFIOD
	A	A	A			A			F			F			F	F

Figura 4.14: "Símbolos y Acrónimos para Dependencias DRBD"
(Fuente: Distefano y Xing, 2006)

Como se puede notar en la figura anterior, cada dependencia se distigue de acuerdo a distintas clasificaciones, que no se verán en detalle dado que posteriores trabajos en ambas líneas de investigación las descartan en su mayoría, como es el caso de los artículos en los que participa Distefano. O éstas clasificaciones son descartadas en su totalidad, caso de los artículos en los que participa Xing. En este éste último, como ya se ha señalado, se alude derechamente a la complejidad de las creaciones.

La línea de investigación trazada por Xing establece en cuanto a modelamiento, tres tipos de dependencias DRBD que se pueden apreciar en la Figura 4.15:

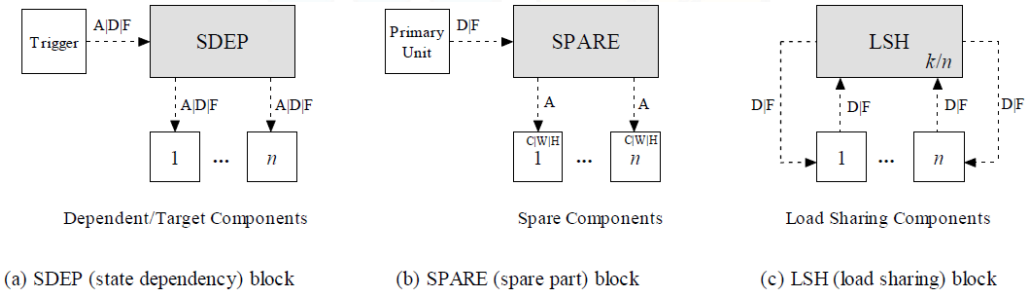


Figura 4.15: "Bloques controladores de componentes DRBD"
(Fuente: Xu et. al., 2008)

Donde A significa activación, D significa desactivación, y F significa falla.

State Dependency Block (SDEP) mostrado en (a) de la Figura 4.15 establece en general que un evento trigger forzará todos los eventos dependientes a ocurrir, Tanto el evento trigger como los eventos dependientes pueden ser activación, desactivación, o falla de una componente del sistema, es decir, SDEP puede modelar nueve tipos de dependencias entre las componentes de un sistema: (A,A), (A,D), (A,F), (D,A), (D,D), (D,F), (F,A), (F,D), y (F,F).

Spare Part Block (SPARE) se muestra en (b) de la Figura 4.15, donde C|W|H se refiere al modo cold|warm|hot en que se encuentra una componente de repuesto. Específicamente, éste bloque modela el comportamiento en que la desactivación o la falla de una componente primaria llevará a la activación de la primera componente de repuesto; la desactivación o falla de la primera componente de repuesto llevará a la activación de la segunda componente de repuesto, y así en adelante. Todas las unidades de repuesto podrían estar en estado stand by en *cold*, *warm*, o *hot*, y deben ser usadas en orden del 1 a n.

Un ***Load Sharing Block (LSH)***, como aquel ilustrado en (c) de la Figura 4.15, anotado con "*k/n*" se refiere a un controlador con *n* componentes en load sharing, entre las que al menos *k* componentes deben estar acivas o funcionando. Si al mismo tiempo hay exactamente *k* componentes activas, y una de ellas es desactivada o falla, todas las otras componentes activas serán desactivadas, y por lo tanto entrarán en estados *stand by*. Sin embargo, es posible que cuando una de las *k* componentes activas falle, alguna otra componente activa también falle debido a sobrecarga si no es desactivada a tiempo, lo que es representado con la etiqueta "*D*|*F*" en (c) de la Figura 4.15.

Posterior a la definición y ejemplificación de las construcciones DRBD de ésta línea de investigación, se procede a convertir el modelo DRBD a Colored Petri Nets con el objetivo de verificar que el diseño realmente cuenta con las propiedades de comportamiento establecidas. Esto último, una vez hecha la conversión del modelo, se realiza en la práctica con CPN Tool, un programa que admite la edición, simulación, y análisis de Colored Petri Nets, y que incluye un motor de análisis de espacio estados. Como trabajo futuro en Xu et al. (2008) se enfatiza en el desarrollo de un software que pueda traducir automáticamente modelos DRBD a Colored Petri Nets para verificación formal.

Una vez descritas ambas líneas de investigación, en adelante se elige trabajar con el modelo DRBD más general propuesto en Distefano et al. (2006) debido a la simplicidad apreciada en términos gráficos, lo que se debe a que se tiene como base la "dependencia" genérica establecida en los orígenes del modelo DRBD.

4.6. Herramientas de resolución

4.6.1. Metodología de resolución del modelo DRBD

Para el modelo DRBD, según lo mencionado en Distefano y Puliafito (2009), la solución óptima es combinar diferentes técnicas por medio de la adecuada descomposición del modelo DRBD en partes o subsistemas independientes. Las partes involucrando dependencias son identificadas como subsistemas dinámicos. Las otras son identificadas como subsistemas estáticos. Los subsistemas estáticos son analizados usando ecuaciones combinatoriales/analíticas, mientras los subsistemas dinámicos pueden ser analizados con diferentes métodos, de acuerdo a la naturaleza de las distribuciones de confiabilidad y la conplejidad de la parte siendo analizada. Una vez realizada la elaboración separada para cada subsistema, los resultados deben ser unidos aplicando las ecuaciones de estructuras a los subsistemas.

En el mismo trabajo aludido anterioremente se presenta el algoritmo resumido en algunos pasos:

1. *Dividir* el DRBD en tantos subsistemas estáticos y dinámicos como sea posible. Cada subsistema debe ser *independiente* de los otros: no comparte unidades (repetidas) con los otros subsistemas y además no existen dependencias entrantes o salientes hacia/desde él.

2. *Discriminar* los subsistemas independientes dependiendo de si contienen unidades repetidas o dependencias (*dinamica*) o no (*estática*).

3. *Reiterar* el algoritmo desde el paso 1 para todos los *subsistemas dinámicos independientes* identificados en el paso previo, hasta que no sean posibles más descomposiciones.

4. *Seleccionar* para todos los subsistemas independientes, la mejor solución disponible considerando la complejidad de los subsistemas, su topología (simetría, etc), la naturaleza de las funciones de distribución acumuladas especificadas y los tipos de dependencias involucradas.

5. *Analizar hacia atrás* los subsistemas, empezando por el último subsistema identificado por los pasos previos y volviendo recursivamente hasta que todo el sistema es evaluado (*bottom-up*).

4.6.2. Métodos de resolución

En lo que respecta a los subsistemas dinámicos independientes, la solución puede estar dada por diferentes métodos que toman el modelo DRBD y lo traducen a su respectivo dominio de análisis. Los métodos de resolución considerados son los siguientes:

- **Continuous Time Markov Chains (CTMC)**
- **Petri Nets (PN)**
- **Non-Markovian stochastic Petri Nets (NMSPN)**
- **Simulación**

A continuación se define cada uno de dichos métodos:

Continuous Time Markov Chains

De acuerdo a Sutton (2010) las cadenas de Markov son un tipo de análisis estocástico, cuya aplicación muestra todos los posibles estados de un sistema pasando por una serie de transiciones. Cada transición representa una unidad de tiempo en la cual el sistema puede quedarse donde está o puede pasar a un nuevo estado.

Los procesos de Markov son una clase especial de procesos estocásticos. El supuesto básico es que el comportamiento del sistema en cada estado no posee memoria. Esto significa que la transición desde el estado actual de un sistema es determinada solo por éste estado y no por sus estados previos o el tiempo al que el estado presente es alcanzado. (Dubrova, 2013)

Antes que la transición ocurra, el tiempo gastado en cada estado sigue una distribución exponencial. Este supuesto es satisfecho si todos los eventos (fallas, reparaciones, etc.) suceden a una tasa de ocurrencia constante. Debido a este supuesto, los procesos de Markov

no pueden ser usados para modelar el comportamiento de sistemas que son sujetos a desgaste de componentes. En cambio se deberían usar General stochastic processes. Los procesos de Markov son clasificados de acuerdo a características de espacio de estados (state space) y espacio de tiempo (time-space) como se muestra en la Figura 4.16.

State space	Time space	Common model name
Discrete	Discrete	Discrete time Markov chains
Discrete	Continuous	Continuous time Markov chains
Continuous	Discrete	Continuous state, discrete time Markov processes
Continuous	Continuous	Continuous state, continuous time Markov processes

Figura 4.16: "Tipos de procesos de Markov"
(Fuente: Fault-Tolerant Design)

En la mayoría de las aplicaciones de análisis de fiabilidad (dependability, concepto que resume las características de seguridad, confiabilidad, y disponibilidad), el espacio de estados es discreto. Por ejemplo, cada componente de un sistema puede tener dos estados: operacional o fallado. La escala del tiempo es usualmente continua, lo cual significa que la falla de la componente y el tiempo de reparación son variables aleatorias. Por lo tanto, Continuous Time Markov Chains (también llamadas Modelos de Markov Continuos) son las más comunes.

Los procesos de Markov son ilustrados graficamente por diagramas de transición de estados. Un diagrama de transición de estados es un gráfico dirigido $G = (V, E)$, donde V es el conjunto de vértices representando los estados del sistema y E es el conjunto de aristas representando las transiciones del sistema. Para modelos de fiabilidad, un estado es definido por la combinación particular de componentes operativas y falladas. Por ejemplo, si un sistema es compuesto por dos componentes, entonces existen 4 combinaciones diferentes enumeradas en la Figura 4.17, donde O indica una componente operacional y F indica una componente fallada.

Las transiciones de estados reflejan los cambios que ocurren en el estado del sistema. Por ejemplo, si un sistema con dos componentes idénticas se encuentra en el estado (OO) y el primer módulo falla, el estado del sistema cambia a (FO). Entonces, un proceso de

Component		State
1	2	Number
O	*O*	1
O	*F*	2
F	*O*	3
F	*F*	4

Figura 4.17: "Estados de una cadena de Markov de dos componentes"
(Fuente: Fault-Tolerant Design)

Markov representa posibles cadenas de eventos que ocurren dentro del sistema. En el caso de evaluación de fiabilidad, estos eventos son fallas y reparaciones.

Cada arista del diagrama de transición de estado lleva una etiqueta, reflejando la tasa a la cual la transición de estado ocurre. Dependiendo de los objetivos del modelamiento, se puede tratar de una tasa de falla o una tasa de reparación.

Un sistema de una componente tiene solo dos estados: operacional (estado 1) y fallado (estado 2). Si no son permitidas las reparaciones, hay una sola transición, no reversible, entre los estados, con la etiqueta λ correspondiente a la tasa de falla de la componente. Si la reparación es permitida, la transición entre los estados fallado y operacional es posible, con la etiqueta μ correspondiente a la tasa de reparación de la componente. Lo anterior se puede apreciar en la Figura 4.18.

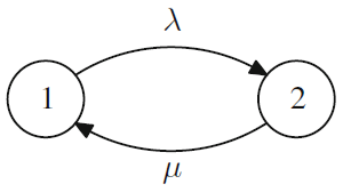


Figura 4.18: "Cadena de Markov de un sistema de una componente"
(Fuente: Fault-Tolerant Design)

En un sistema de 2 componentes son posibles 4 estados, enumerados en la Figura 4.17. Los cambios de estados son representados por el diagrama de transiciones de estados de la Figura 4.19. Las tasas de falla λ_1 y λ_2 de las componentes 1 y 2, respectivamente, indican la tasa a la cual las transiciones son hechas entre los estados. Se asume que las dos componentes son independientes y no reparables.

Si las componentes se encuentran en configuración de serie. entondes cualquier falla

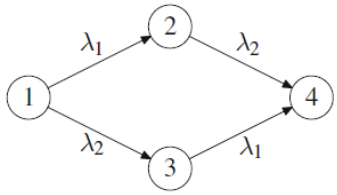


Figura 4.19: "Cadena de Markov de un sistema de dos componentes, sin reparación"
(Fuente: Fault-Tolerant Design)

de una componente causa la falla del sistema. Entondes, solo el estado 1 es un estado operacional. Los estados 2, 3, y 4 son estados fallados. Si las componentes se encuentran en configuración paralelo, ambas deben fallar para que el sistema falle. por lo tanto, los estados 1, 2, y 3 son estados operacionales, mientras el estado 4 es un estado fallado.

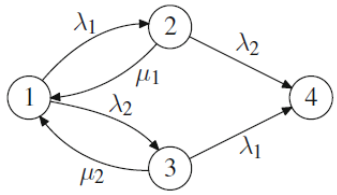


Figura 4.20: "Cadena de Markov de un sistema de dos componentes, con reparación"
(Fuente: Fault-Tolerant Design)

Si se asume que las componentes se encuentran en paralelo y pueden ser reparadas mientras el sistema no falle, es decir, que las componentes pueden ser reparadas en el estado 2 y 3 mostrados en la Figura 4.17, el sistema queda representado en la Figura 4.20, donde μ_1 y μ_2 son las tasas de reparación de las componentes 2 y 1 respectivamente.

Frecuentemente es posible reducir el tamaño del diagrama de transición de estado sin sacrificar la exactitud del análisis. Por ejemplo, si los componentes del sistema de dos componentes representado por el diagrama de transición de estados de la Figura 4.19 tienen tasa de falla idéntica $\lambda_1 = \lambda_2 = \lambda$, no es necesario distinguir entre los estados 2 y 3. Ambos estados representan una condición donde una componente esta operacional y una esta fallada. Entonces se pueden unir estos dos estados dentro de uno solo.

Dado que las fallas de las componentes se asumen como eventos independientes, la tasa de transición desde el estado 1 al 2 es la suma de las tasas de transición desde los estados 1 hacia el 2 y el 3, es decir, 2λ.

Un ejemplo de modelamiento con cadenas de Markov se puede realizar considerando

el siguiente modelo RBD:

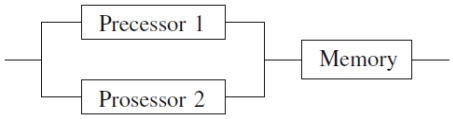


Figura 4.21: "Modelo RBD, ejemplo"
(Fuente: Fault-Tolerant Design)

Como se puede apreciar en la Figura 4.21 el sistema funciona cuando funciona la memoria y uno de los dos procesadores. Con esto se genera la siguiente información:

Component			State
P_1	P_2	M	Number
O	O	O	1
O	O	F	3
O	F	O	2
O	F	F	3
F	O	O	2
F	O	F	3
F	F	O	3
F	F	F	3

Figura 4.22: "Descripción de estados, ejemplo"
(Fuente: Fault-Tolerant Design)

A partir de la Figura 4.22 se puede generar la siguiente cadena de markov:

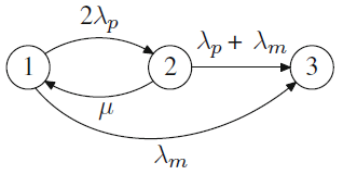


Figura 4.23: "Cadena de Markov, ejemplo"
(Fuente: Fault-Tolerant Design)

Las cadenas de Markov también se pueden usar para evaluar fiabilidad (dependability), dentro de ellas la clase de procesos de Markov más importante para dicho objetivo son Continuous time markov chains.

El objetivo del análisis de los procesos de Markov es calcular $P_i(t)$, la probabilidad de que el sistema se encuentre en el estado i en el tiempo t. Una vez esto es conocido, la confiabilidad, disponibilidad y seguridad puede ser calculada como la suma de las probabilidades de todos los estados operacionales.

Suponiendo que el estado 1 es el estado en el que todos las componentes se encuentran operacionales. Asumiendo que en $t = 0$ el sistema esta en el estado 1, se obtiene:

$$P_1(0) = 1 \tag{4.11}$$

Dado que en cualquier tiempo el sistema puede estar solo en un estado, $P_i(0) = 0$, $\forall i \neq 1$, y se obtiene:

$$\sum_{i \in O \cup F} P_i(t) = 1 \tag{4.12}$$

Donde la suma es sobre todos los posibles estados operacionales y fallados.

Para determinar las probabilidades $P_i(t)$, se derivan un conjunto de ecuaciones diferenciales, una por cada estado i del sistema. Estas ecuaciones son llamadas *ecuaciones de transición de estado*, porque permiten determinar las probabilidades $P_i(t)$ en términos de las tasas (de falla o reparación) a las cuales son realizadas las transiciones desde un estado a otro. Las ecuaciones de transición de estados usualmente son presentadas en forma de matriz. La matriz M cuya entrada m_{ij} es la tasa de transición entre los estados i y j es llamada la matriz de transición. Se usa el primer índice i para las columnas de la matriz y el segundo índice j para las filas, es decir, M tiene la siguiente estructura:

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{21} & \dots & m_{k1} \\ m_{12} & m_{22} & \dots & m_{k1} \\ & & \dots & \\ m_{1k} & m_{2k} & \dots & m_{kk} \end{bmatrix}$$

Figura 4.24: "Matriz de transición"
(Fuente: Fault-Tolerant Design)

Donde k es el número de estados en una cadena de Markov representativa del sistema. Las entradas m_{ii} correspondientes a auto transiciones son calculadas como $-\sum m_{ij}, \forall i, j \in 1, 2, ..., k$ tal que $i \neq j$. En consecuencia, las entradas en cada columna de la matriz de transición suma cero. Un signo positivo en una entrada ij-ésima indica que la transición se origina en el estado i-ésimo. Un signo negativo en una entrada ij-ésima indica que la trnasición termina en el estado i-ésimo.

En la evaluación de confiabilidad, una vez el sistema falla, el estado fallado no puede ser dejado. Por lo tanto, se puede usar la matriz de transición para distinguir entre el estado fallado y operacional. Cada estado fallado i tiene un elemento diagonal cero m_{ii}. No es el caso de cuando se calcula la disponibilidad y la seguridad.

Utilizando matrices de transición, las ecuaciones de transición de estados se derivan de la siguiente forma. Sea $P(t)$ un vector cuyo elemento i-ésimo es la probabilidad $P_i(t)$ del sistema en el estado i al tiempo t. Así la matriz representación del sistema de las ecuaciones de transición de estados queda dada por:

$$\frac{d}{dt}P(t) = M \times P(t) \tag{4.13}$$

Una vez el sistema de ecuaciones es resuelto y las probabilidades $P_i(t)$ son conocidas, la confiabilidad, disponibilidad, o seguridad pueden ser calculadas como la suma de las probabilidades considerando todos los estados operacionales O:

$$R(t) = \sum_{i \in O} P_i(t) \tag{4.14}$$

o, alternativamente:

$$R(t) = 1 - \sum_{i \in F} P_i(t) \tag{4.15}$$

Donde la suma considera todos los estados fallados F.

Evaluación de confiabilidad

Existen diferentes procedimientos para calcular confiabilidad dependiendo de si se trata de componentes independientes o dependientes.

Componentes independientes

En un sistema compuesto por dos componentes en paralelo, que no son reparables, se tiene la siguiente matriz de transición:

$$M = \begin{bmatrix} -2\lambda & 0 & 0 \\ 2\lambda & -\lambda & 0 \\ 0 & \lambda & 0 \end{bmatrix} \tag{4.16}$$

Luego, para obtener las ecuaciones de transisición de estados del sistema se tiene que:

$$\frac{d}{dt}\begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \end{bmatrix} = \begin{bmatrix} -2\lambda & 0 & 0 \\ 2\lambda & -\lambda & 0 \\ 0 & \lambda & 0 \end{bmatrix} \times \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \end{bmatrix} \tag{4.17}$$

Obteniendo las siguientes ecuaciones diferenciales:

$$\begin{cases} \frac{d}{dt}P_1(t) = -2\lambda P_1(t) \\ \frac{d}{dt}P_2(t) = 2\lambda P_1(t) - \lambda P_2(t) \\ \frac{d}{dt}P_3(t) = \lambda P_2(t) \end{cases} \tag{4.18}$$

Resolviendo el sistema de ecuaciones se tiene:

$$\begin{aligned} P_1(t) &= e^{-2\lambda t} \\ P_2(t) &= 2e^{-\lambda t} - 2e^{-2\lambda t} \\ P_3(t) &= 1 - 2e^{-\lambda t} + e^{-2\lambda t} \end{aligned} \tag{4.19}$$

Dado que son conocidas las probabilidades $P_i(t)$ se puede calcular la confiabilidad del sistema (que se encuentra en paralelo) como la suma de las probabilidades $P_1(t)$ y $P_2(t)$:

$$R_{paralelo}(t) = 2e^{-\lambda t} - e^{-2\lambda t} \tag{4.20}$$

Considerando la ecuación de estructura en paralelo y que la confiabilidad de una componente es $R(t) = e^{-\lambda t}$, se obtiene que:

$$R_{paralelo}(t) = R(t)^2 + 2R(t) * (1 - R(t)) = e^{-2\lambda t} + 2e^{-\lambda t}(1 - e^{-\lambda t}) = 2e^{-\lambda t} - e^{-2\lambda t} \tag{4.21}$$

Dado lo anterior se reafirma que los resultados del análisis con cadenas de Markov y RBD son los mismos, y esto se debe al supuesto de que las fallas de las componentes son mutuamente independientes.

Componentes dependientes

El valor de los procesos de Markov se vuelve evidente en situaciones en las cuales las tasas de falla de las componentes dependen del estado del sistema. Un ejemplo típico son las componentes en load sharing y en stand by.

La palabra carga (load) se usa en un sentido amplio.Puede ser una carga eléctrica, una carga causada por alta temperatura, o una carga de información. En la práctica, la tasa de falla tipicamente se incrementa cuando la carga se aumenta. Suponiendo que varias componentes comparten una carga, si una componente falla, la carga adicional en la componente restante es probable que incremente su tasa de falla.

Un ejemplo de un sistema de dos componentes en load sharing queda representado en el siguiente diagrama de transición de estados:

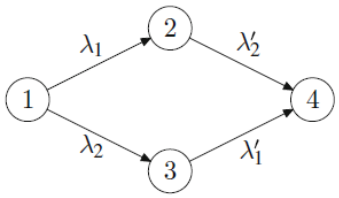


Figura 4.25: "Diagrama de transición de estados, sistema load sharing de dos componentes"
(Fuente: Fault-Tolerant Design)

Los estados quedan descritos de acuerdo a la Figura 4.17. Luego de una falla de la primera componente, la tasa de falla de la segunda componente se incrementa. La tasa de falla incrementada de las componentes 1 y 2 son denotadas por λ_1' y λ_2', respectivamente.

Para derivar las ecuaciones de transición de estados se tiene lo siguiente:

$$\frac{d}{dt}\begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix} = \begin{bmatrix} (-\lambda_1 - \lambda_2) & 0 & 0 & 0 \\ \lambda_1 & -\lambda_2' & 0 & 0 \\ \lambda_2 & 0 & -\lambda_1' & 0 \\ 0 & \lambda_2' & \lambda_1' & 0 \end{bmatrix} \times \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix} \tag{4.22}$$

Se obtienen las siguientes ecuaciones diferenciales:

$$\begin{cases} \frac{d}{dt}P_1(t) = (-\lambda_1 - \lambda_2)P_1(t) \\ \frac{d}{dt}P_2(t) = \lambda_1 P_1(t) - \lambda_2^{'} P_2(t) \\ \frac{d}{dt}P_3(t) = \lambda_2 P_1(t) - \lambda_1^{'} P_3(t) \\ \frac{d}{dt}P_4(t) = \lambda_2^{'} P_2(t) + \lambda_1^{'} P_3(t) \end{cases} \tag{4.23}$$

Resolviendo el sistema de ecuaciones se tiene:

$$\begin{aligned} P_1(t) &= e^{(-\lambda_1-\lambda_2)t} \\ P_2(t) &= \frac{\lambda_1}{\lambda_1+\lambda_2-\lambda_2^{'}} e^{\lambda_2^{'}t} - \frac{\lambda_1}{\lambda_1+\lambda_2-\lambda_2^{'}} e^{(-\lambda_1-\lambda_2)t} \\ P_3(t) &= \frac{\lambda_2}{\lambda_1+\lambda_2-\lambda_1^{'}} e^{\lambda_1^{'}t} - \frac{\lambda_2}{\lambda_1+\lambda_2-\lambda_1^{'}} e^{(-\lambda_1-\lambda_2)t} \\ P_4(t) = 1 - e^{(-\lambda_1-\lambda_2)t} &- \frac{\lambda_1}{\lambda_1+\lambda_2-\lambda_2^{'}} e^{\lambda_2^{'}t} + \frac{\lambda_1}{\lambda_1+\lambda_2-\lambda_2^{'}} e^{(-\lambda_1-\lambda_2)t} \\ &- \frac{\lambda_2}{\lambda_1+\lambda_2-\lambda_1^{'}} e^{\lambda_1^{'}t} + \frac{\lambda_2}{\lambda_1+\lambda_2-\lambda_1^{'}} e^{(-\lambda_1-\lambda_2)t} \end{aligned} \tag{4.24}$$

Finalmente, dado que el estado $P_4(t)$ es el estado fallado, la confiabilidad del sistema es igual a $1 - P_4(t)$:

$$\begin{aligned} R_{paralelo}(t) = e^{(-\lambda_1-\lambda_2)t} &+ \frac{\lambda_1}{\lambda_1+\lambda_2-\lambda_2^{'}} e^{\lambda_2^{'}t} - \frac{\lambda_1}{\lambda_1+\lambda_2-\lambda_2^{'}} e^{(-\lambda_1-\lambda_2)t} \\ &+ \frac{\lambda_2}{\lambda_1+\lambda_2-\lambda_1^{'}} e^{\lambda_1^{'}t} - \frac{\lambda_2}{\lambda_1+\lambda_2-\lambda_1^{'}} e^{(-\lambda_1-\lambda_2)t} \end{aligned} \tag{4.25}$$

Si $\lambda_1^{'} = \lambda_1$ y $\lambda_2^{'} = \lambda_2$ la confiabilidad del sistema queda como el caso de las componentes independientes. Por otra parte, se puede simplificar el resultado si las componentes fueran idénticas (misma tasa de falla).

Petri Nets

Una Petri net (PN) es un gráfico dirigido bipartito, compuesto por dos tipos de nodos, llamados *lugares*, P, y *transiciones*, T. Los arcos entre los nodos caen en dos categorías: *arcos de entrada* que conducen desde un lugar de entrada a una transición, y *arcos de salida*

que conducen de una transición a un lugar de salida. Los arcos siempre tienen que ir desde un tipo de nodo al complementario. Un número finito de *marcas* pueden ser distribuidas entre los lugares y cada lugar puede contener un número arbitrario (natural) de marcas. Un *marcado* $m \in \mathcal{M}$ se define como una posible asignación de marcas a todos los lugares en la PN. Los marcados son algunas veces referidos como *estados* de la PN. Si P denota el conjunto de lugares, entonces un marcado m es un conjunto múltiple, $m \in \mathcal{M} \subset \mathbb{N}^{|P|}$, que describe el número de marcas en cada lugar. Para referirse al número de marcas de un lugar particular $P_i, 1 \leq i \leq |P|$, en un marcado m, se utiliza la notación $\#(P_i, m)$. (Bolch et al., 2006)

Una transición se habilita si todos sus lugares de entrada contienen al menos una marca. Notar que una transición sin arcos de entrada esta siempre habilitada. Una transición habilitada puede disparar removiendo una marca de cada lugar de entrada y agregando una marca a cada lugar de salida. El disparo de una transición puede transformar una PN desde un marcado a otro. Con respecto a un marcado inicial dado, el *conjunto de alcanzabilidad*, $\mathcal{RS}$, se define como el conjunto de todos los marcados alcanzados a través de cualquier secuencia de disparo posible de transiciones, empezando desde el marcado inicial. El $\mathcal{RS}$ podría ser infinito si no se imponen restricciones. Incluso PNs aparentemente simples pueden dar lugar a $\mathcal{RS}s$ grandes o infinitos. Si más de una transición es habilitada simultáneamente y no pueden dispararse en paralelo, se presenta un conflicto entre transiciones que debe ser resuelto por medio de una política de selección, por ejemplo, basada en un esquema de prioridad. Los marcados en los cuales ninguna transición es habilitada son llamados marcados absorbentes.

Para una representación gráfica, los lugares usualmente son simbolizados por círculos y las transiciones por barras. Una transición simple con un marcado inicial:

$$m_0 = (2, 0, 0, 0), \tag{4.26}$$

y con marcado subsiguiente

$$m_1 = (1, 1, 1, 0), \tag{4.27}$$

alcanzable por medio del disparo de la transición t_1, se muestra en la Figura 4.26:

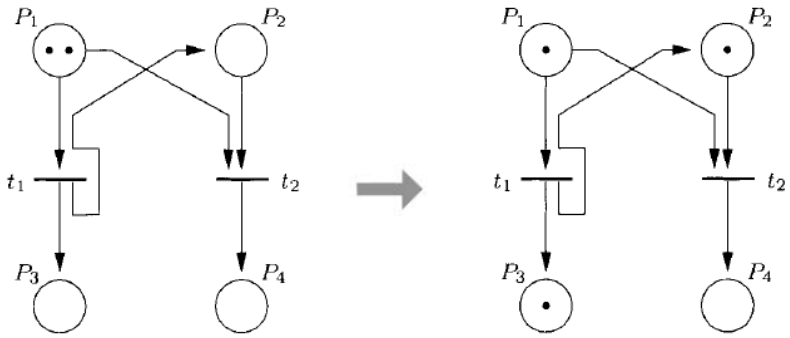


Figura 4.26: "PN simple"
(Fuente: Queueing networks and markov chains)

La transición t_1 es habilitada porque su único lugar de entrada P_1 contiene dos marcas. Dado que P_2 está vacío, la transición t_2 está deshabilitada. Cuando el disparo sucede, una marca es removida desde el lugar de entrada P_1 y una marca es depositada en ambos lugares de salida P_2 y P_3. En el nuevo marcado, ambas transiciones t_1 y t_2 son habilitadas. El conflicto aparente entre t_1 y t_2 debe ser resuelto para decidir cual de los marcados alternativos (absorbente) será alcanzado:

$$m_2 = (0, 2, 2, 0) \quad o \quad m_3 = (0, 0, 1, 1) \tag{4.28}$$

El conjunto de alcanzabilidad en éste ejemplo estará dado por $\{ m_0, m_1, m_2, m_3 \}$, donde los marcados estan dados según lo definido anteriormente.

Una PN esta dada por una 5-tupla, $PN = \{P, T, D^-, D^+, m_0\}$ con:

- Un conjunto finito de lugares $P = \{P_1, P_2, ..., P_{|P|}\}$, donde cada lugar contiene un número no negativo de marcas.
- Un conjunto finito de transiciones $\{ T = t_1, ..., t_{|T|} \}$, tal que $T \cap P = \emptyset$.
- Un conjunto de arcos de entrada $D^- \in \{ 0, 1 \}^{|P \times T|}$ y el conjunto de arcos de salida $D^+ \in \{ 0, 1 \}^{|P \times T|}$. Si $D^-(P_i, t_k) = 1$, entonces existe un arco de entrada desde el lugar P_i a la transición t_k, y si $D^+(P_j, t_l) = 1$, entonces existe un arco de salida desde la transición t_l al lugar P_j.

- Un marcado inicial $m_0 \in \mathcal{M}$.

Ejemplo de Petri net en sistema de fabricación

En Hrúz y Zhou (2007) se puede encontrar un ejemplo de Petri net aplicado a un sistema de manufactura. En éste se considera la descripción de la Figura 4.27, en que una pieza que llega al centro de trabajo a partir del transportador de entrada (*input conveyor*) es transferida a la máquina de molienda (*milling machine*), luego, el robot y la pieza deben estar disponibles para comenzar la operación de transferencia a la entrada de la máquina. Posterior a ello se realiza la molienda con la máquina de molienda, que debe estar disponible para ello. Luego de la molienda, la pieza se mueve hacia la salida de la máquina. Si el robot se encuentra disponible se realiza la transferencia de la pieza hacia el transportador de salida (*output conveyor*).

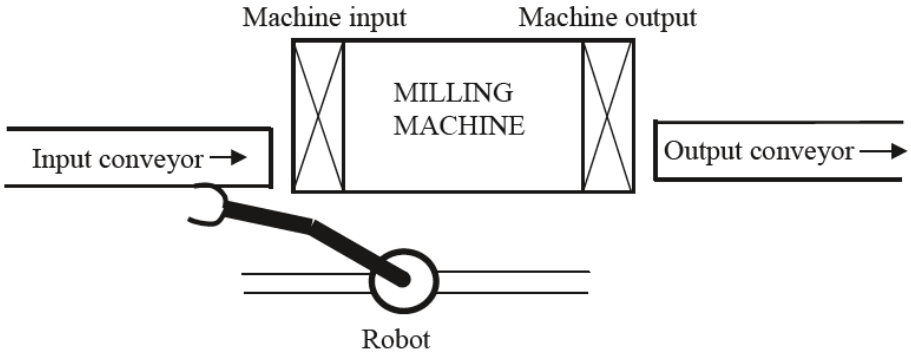


Figura 4.27: "Sistema de manufactura"
(Fuente: Modelling and control of discrete-event dynamic systems)

La Petri net que modela el sistema descrito anteriormente se puede apreciar en la siguiente figura:

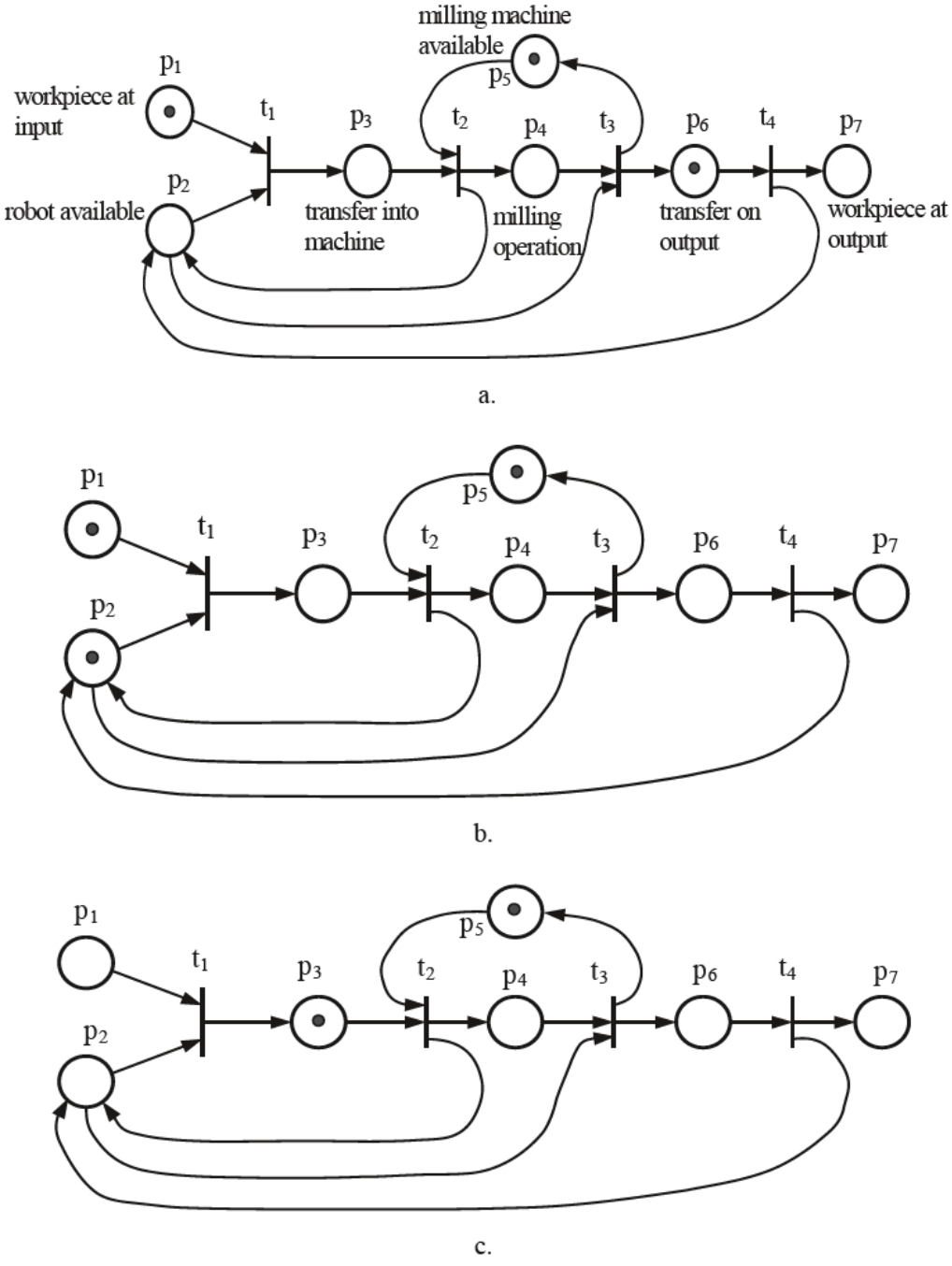


Figura 4.28: "Petri net de un sistema de manufactura"
(Fuente: Modelling and control of discrete-event dynamic systems)

La información de la Figura 4.28 se detalla en la siguiente tabla:

Tabla 4.1: Detalle de lugares y transiciones, Petri net de sistema de manufactura.

Notación	Tipo	Descripción
p_1	Lugar, susbsistema	Transportador de entrada
p_2	Lugar, susbsistema	Robot
p_5	Lugar, susbsistema	Máquina de molienda
p_7	Lugar, susbsistema	Transportador de salida
p_3	Lugar, operación	Transferencia a la entrada de la máquina de molienda
p_4	Lugar, operación	Molienda
p_6	Lugar, operación	Transferencia a la salida de la máquina de molienda
t_1	Transición	Comienzo de la transferencia de entrada a la máquina de molienda
t_2	Transición	Término de la transferencia de entrada a la máquina de molienda y comienzo de la molienda
t_3	Transición	Término de la molienda y comienzo del transporte de salida
t_4	Transición	Término de la transferencia de salida

En a de la Figura 4.28 se puede apreciar que las condiciones para la transferencia de la pieza a la máquina de molienda no se satisfacen por lo cual no se activa la transición. En b de la Figura 4.28 se muestra cuando las condiciones si se satisfacen, luego en c se dispara la transición dejando una marca en p_3 . Finalmente en la Figura 4.29 se muestra cuando se comienza la operación de molienda.

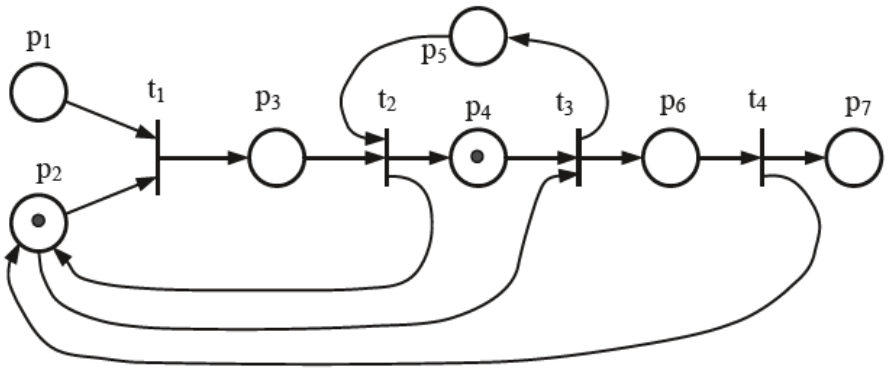


Figura 4.29: "Petri net de un sistema de manufactura, molienda"
(Fuente: Modelling and control of discrete-event dynamic systems)

De acuerdo con Volovoi (2004) las Petri nets originales no incluyen el tiempo, por lo cual las transiciones se habilitan en forma instantánea. La introducción de tiempos de retraso en forma determinística (donde los disparos de las transiciones ocurren en un tiempo específico), da lugar a las Petri nets temporizadas (*Timed Petri nets*). Si estos tiempos de retraso son variables aleatorias basados en distribuciones dadas, se tiene una Petri net estocástica (*SPN, Stochastic Petri Net*).

De acuerdo a lo anterior y en consideración de lo señalado en Bolch et al. (2006), existen las GSPNs que generalizan PNs de tal manera que cada transición tiene un *tiempo de disparo* asignado a ella, lo cual puede ser exponencialmente distribuido o cero constante. Las transiciones con tiempos de disparo exponencialmente distribuidos son llamadas transiciones *temporizadas*, mientras las otras son referidas como transiciones *inmediatas*. En consecuencia el nuevo tipo de transiciones requiere reglas de habilitado y de disparo para ser adaptadas. (Bolch et al., 2006)

Los marcados $\mathcal{M} = \mathcal{V} \cup \mathcal{T}$ en el conjunto de alcanzabilidad $\mathcal{RS}$ de una GSPN son particionados en dos conjuntos, el conjunto de marcados *desvanecidos* $\mathcal{V}$ y el conjunto de marcados *tangibles* $\mathcal{T}$. Los marcados desvanecidos comprenden aquellos en que al menos una transición inmediata es habilitada. Si ninguna transición inmediata es habilitada, es decir, solo se habilitan transiciones temporizadas o no hay transiciones, resulta un marcado tangible. Los marcados desvanecidos no residen en ningún tiempo finito distinto de cero y los disparos son realizados instantáneamente en este caso. Por lo tanto, las transiciones inmediatas siempre tienen prioridad sobre las transiciones temporizadas para dispararse. Si varias transiciones inmediatas compiten por dispararse, una función de probabilidad es usada para romper éste empate. Si las transiciones temporizadas compiten, un modelo de competencia es aplicado de modo que la transición cuyo tiempo de disparo transcurre primero es la siguiente en disparar.

Desde una GSPN dada, un *gráfico de alcanzabilidad extendido* ($\mathcal{ERG}$) es generado conteniendo los marcados del conjunto de alcanzabilidad como nodos y alguna información estocástica adjunta a los arcos, relacionando de este modo las marcas entre sí. Los circuitos absorbentes de marcados desvanecidos son a priori excluidos desde el $\mathcal{ERG}$, porque de otro modo podría resultar una discontinuidad estocástica. Las GSPNs que interesan son

limitadas, es decir, el conjunto de alcanzabilidad subyacente es finito.

Barras gruesas son usadas para representar graficamente transiciones temporizadas, mientras barras delgadas son reservadas para transiciones inmediatas. Un ejemplo de una GSPN se muestra en la Figura 4.30. En contraste con una PN simple, un tiempo de disparo exponencialmente distribuido con tasa λ_1 ha sido asignado a la transición t_1. Mientras en una PN simple de la derecha de la Figura 4.26 ambas transiciones t_1 y t_2 se encuentran habilitadas, solo la transición inmediata t_2 se encuentra habilitada en la GSPN de la derecha de la Figura 4.30, porque las transiciones inmediatas tienen prioridad sobre las transiciones temporizadas.

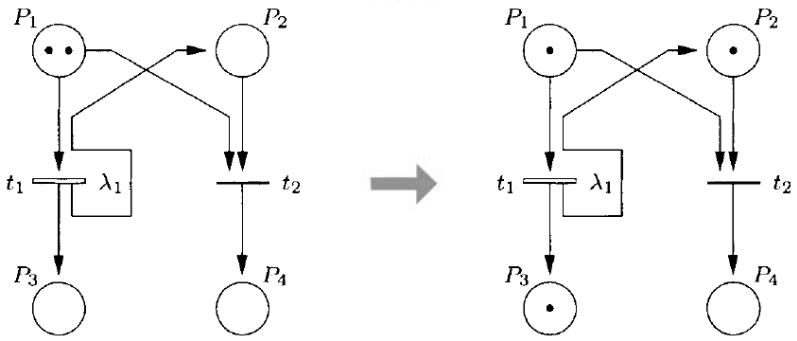


Figura 4.30: "Ejemplo de GSPN"
(Fuente: Queueing networks and markov chains)

En adelante se mencionará a las Petri Nets refiriendose a las Generalized Stochastic Petri Nets descritas previamente, considerando que éstas últimas son su última versión.

En cuanto al estudio de la confiabilidad, una Petri net considerará una serie de sensores que perciban los cambios que puedan ocurrir, elementos que serán reflejados en las transiciones. Estos sensores estarían programados para detectar condiciones específicas como el alza de temperatura en una caldera o el exceso de fluidos en una bomba centrífuga, a la vez que se podrían integrar como condiciones de transición la ocurrencia de un evento como los descritos para el modelamiento DRBD (falla, reparación, dehabilitación, y habilitación de una componente). Debido a lo anterior las Petri nets poseen un alto poder para describir en forma teórica el comportamiento de un sistema, sin embargo para posteriores cambios se debe modificar constantemente el modelamiento realizado.

Non-Markovian stochastic Petri Nets

"Petri nets representan una herramienta útil para análisis de rendimiento, fiabilidad, y capacidad de ejecución de sistemas complejos. Su poder de modelamiento puede ser incrementado incluso más si se considera que los eventos no son exponencialmente distribuidos." (Bobbio et al., 2000)

La inclusión de distribuciones no exponenciales implica que el sistema no posee la propiedad markoviana, y de ésto se desprende su denominación.

En Rogge-solti y Weske (2015) se hace mención a la ventaja de éste tipo de método, que es la mayor precisión en los resultados.

Simulación

"La simulación es el proceso de diseñar y desarrollar un modelo computarizado de un sistema o proceso y realizar pruebas con este modelo con el propósito de entender el comportamiento del sistema o evaluar estratégias con las cuales se puede operar el sistema." (Melo et al., 2009)

Una forma de simulación conocida es la simulación Monte Carlo. Como lo detalla brevemente Sutton (2010) el método funciona basicamente de la siguiente forma:

1. A cada ítem del equipamiento (y operador humano) se le asigna una probabilidad de falla y una condición inicial de operación.

2. Se inicia el reloj de la computadora en el tiempo cero. Cada "tic" del reloj de la computadora representa un intervalo de tiempo, tal como 12 horas, de la operación actual.

3. Un número aleatorio se genera para cada tiempo en el sistema.

4. Si el número aleatorio cae dentro de un rango predefinido, entonces el ítem es puesto en un estado fallado.

5. Al final de cada iteración un examen de los conjuntos de cortes determina la operabilidad global del sistema y la disponibilidad de recursos.

6. Si un ítem falla durante la iteración actual es dejado fuera del servicio correspondiente al tiempo de reparación y la disponibilidad de recursos de mantenimiento.

7. Las iteraciones son continuadas hasta que se obtiene una representación global estable de la disponibilidad del sistema.

Generadores de números aleatorios

En el centro de cualquier modelo estocástico se encuentra un generador de números aleatorios. Este es un algoritomo que genera una serie de números de modo que cada número sucesivo tiene una igual probabilidad de poseer cualquier valor, y cada número es estadísticamente independiente de los otros números de la serie. En la vida real la aleatoriedad no existe - cada consecuencia tiene una causa. Sin embargo, en muchos casos la relación de causa-consecuencia no se conoce, entonces una secuencia aleatoria de eventos es usada para simular el mundo real.

Ningún algoritmo matemático puede generar un conjunto de números realmente aleatorios porque los cálculos y respuestas de los algoritmos son, en principio, completamente predecibles. No obstante, un algoritmo puede generar números pseudo-aleatorios, es decir, series de números que no poseen un patrón discernible.

Un algoritmo de números aleatorios se inicia con un valor semilla, y de éste genera un segundo número. El segundo número luego es usado como semilla de un tercer número, y así en adelante.

Generalmente ecuaciones de números aleatorios son de la siguiente forma:

$$X_{n+1} = modulus((a * X_n + b)/c) \tag{4.29}$$

Donde a, b, y c son números no negativos, y X_n es un número en la serie de números aleatorios. (Modulus retorna el resto luego de la división de un número por otro. Entonces mod(9/9) es 0, mod(8/9) es 8, y mod (100/9) es 1).

La dificultad con cualquier generador de números aleatorios es que es frecuente algún ritmo o patrón en los números que empieza a ser significativo si la secuencia de numeración es mantenida un tiempo suficiente. Dicho patrón puede ser difícil de detectar, pero siempre

estará presente, una vez un número es repetido, el ciclo es reiniciado. Un método de abordar este problema es reiniciar la secuencia ocasionalmente con un nuevo valor semilla, y posiblemente con un nuevo algoritmo.

La mayoría de los compiladores del lenguaje de la computadora incluyen un generador de números aleatorios. Antes de usar uno de estos es importante establecer que el algoritmo genera una secuencia de números con un alto grado de aleatoriedad.

Acelerando la simulación

Uno de los mayores inconvenientes de la simulación Monte Carlo es el largo tiempo de procesamiento que es frecuentemente necesitado en función de alcanzar resultados estables. En consecuencia es útil usar técnicas que aceleran las simulaciones sin causar deterioro en la calidad de la aleatoriedad de los números usados internamente.

Una forma de acelerar una simulación es parear los números aleatorios generados. Esto se realiza generando un número aleatorio, restando su valor de la unidad (es decir, obteniendo su dual), y luego retornando ambos números como argumentos. El tiempo computacional necesitado para un cálculo simple de resta es mucho menor que el tiempo necesitado para generar un número nuevo, así la simulación procederá mucho más rápido.

En general la simulación puede ser aplicada ampliamente, pero su mayor inconveniente es el alto potencial de desarrollo y costo computacional para obtener resultados certeros.

4.6.3. Comparación de métodos de resolución

Para elegir un método de resolución es necesario generar puntos de comparación entre ellos. En consecuencia, se definen las siguientes características a considerar:

Complejidad. Se refiere a cuán difícil es la utilización del método. Se puede encontrar influenciado por el nivel de definición que existe del mismo dentro de la comunidad científica (artículos, libros, web, etc.), y puede acarrear mayor tiempo de procesamiento.

Aplicabilidad. Se define como la capacidad del método para ser utilizado en confiabilidad, que es la métrica de interés en éste trabajo.

Precisión. Se define como el nivel de exactitud de los resultados de acuerdo a lo mencionado por diferentes autores. Se verá influenciado por lo apropiado de los supuestos realizados.

A continuación se dará a conocer el análisis en base a cada característica para los diferentes métodos definidos previamente:

Complejidad

Las CTMC presentan un lenguaje simple de modelamiento que requiere un análisis previo en relación a definición de estados, siempre teniendo presente que se modela en función de obtener confiabilidad (caso distinto es para la obtención de disponibilidad y seguridad). El posterior procesamiento con ecuaciones diferenciales para obtener resultados de probabilidades de encontrarse un estado en particular es un procedimiento conocido que tiene amplio nivel de definición con innumerables referencias.

Las Petri nets requieren de un mayor manejo de detalles técnicos referentes a su modelamiento mismo, lo que entorpece el enfoque de resolver el sistema en cuanto a confiabilidad. Cabe mencionar que son un método relativamente definido y cierto grado de consenso en distintas fuentes.

Las Petri Nets estocásticas no-Markovianas son un método que aún se encuentra en estudio, no pareciendo existir total consenso en la especificación práctica del mismo.

La simulación es un método atractivo por el grado de precisión que se podría alcanzar en los resultados, sin embargo, puede significar un costo en cuanto a tiempo de desarrollo de un modelo de simulación particular para resolver un problema. Por otra parte existen diferentes tipos de simulación que deben ser estudiados para elegir en base a conveniencia el que se aplicará. El tiempo computacional podría llegar a ser alto dependiendo de la complejidad del sistema y de las herramientas computacionales utilizadas.

Aplicabilidad

De acuerdo a lo visto en los ejemplos las CTMC son altamente aplicables a confiabilidad, sin embargo, el problema “state space explosion”, que es cuando el espacio de estados se

vuelve muy grande o incluso infinito (especialmente si los equipos no se suponen idénticos), es algo con lo que se podría tener que lidiar, en cuyo caso es necesario contar con técnicas de reducción (si procede) que permitan hacer el problema más manejable.

Dado que las Petri Nets incluyen la caracterización no solo de los estados del sistema, sino también de la secuencia lógica de actividades y de las componentes que determinan los cambios del sistema, se requiere de cierta dedicación previa determinar ésta información del comportamiento del sistema. Se debe mencionar que su utilización en confiabilidad es ampliamente conocido. Por otra parte, las NMSPN no tienen la misma ponderación en cuanto a ser conocido su uso en confiabilidad, sin embargo, el hecho que se puedan incluir distintos tipos de distribuciones lo convierte en un método promisorio.

La simulación del tipo Monte Carlo es muy conocida como una aplicación en la obtención de confiabilidad de sistemas, ya que, permite modelar cualquier distribución de confiabilidad sin restricciones particulares.

Precisión

La precisión de CTMC puede verse afectada por el mismo supuesto base del método que es la propiedad Markoviana, es decir, que se utilizan distribuciones exponenciales dando como resultado la aplicación de tasas de falla y reparación constantes. Lo anterior en confiabilidad significa que los equipos se encuentran en vida útil, pero también existe el periodo de rodaje y desgaste en el ciclo de vida de un dispositivo modelado usualmente con las distribuciones Weibull y Normal, respectivamente. En consecuencia, la utilización de éste método produce resultados no del todo certeros.

Como las Petri nets poseen la propiedad Markoviana poseen la misma desventaja que las CTMC con respecto a precisión.

En contraste a las alternativas mencionadas en los párrafos precedentes las NMSPN ganan notablemente en cuanto a precisión porque dan lugar a la utilización de distribuciones no solamente exponenciales.

Los resultados de la simulación podrían llegar a tener un alto grado de precisión, dado el mayor grado de libertad en la integración de las características propias del sistema en cuestión.

Resumen

En consecuencia al análisis con respecto a cada característica se presenta la siguiente tabla con la asignación de evaluaciones:

Tabla 4.2: Resumen evaluación de características, métodos de resolución.

Método de resolución	Complejidad	Aplicabilidad	Precisión
CTMC	Baja	Media-Alta	Media
Petri Nets	Media-Baja	Media-Alta	Media
NMSPN	Media	Media	Alta
Simulación	Media	Media-Alta	Alta

Como se puede apreciar de la Tabla 4.2 los métodos que poseen, en general, una mejor evaluación son CTMC y Petri Nets, sin embargo, poseen menor precisión que simulación y NMSPN.

4.6.4. Herramientas computacionales

"Similarmente a lo que ocurre con DFT y otras notaciones de confiabilidad/disponibilidad de alto nivel, DRBD es un formalismo de modelamiento y por lo tanto es necesario recurrir a un motor computacional para resolver un modelo." (Distefano y Puliafito, 2009)

Dado lo anterior, además del método de resolución se debe considerar una herramienta computacional para la obtención de la evaluación final del modelo en términos de confiabilidad.

Como ejemplos de herramientas utilizadas se encuentra en la literatura Xu et al. (2008) el uso de CPN Tool, que es un programa que permite edición, simulación, y análisis de colored Petri Nets. Otras herramientas que se mencionan Distefano y Puliafito (2009) son WebSPN Tool que permite analizar Stochastic Generalized Petri Nets (GSPNs), y DRPFTproc tool para analizar Petri Nets.

5 | PROPUESTA METODOLÓGICA

Dado el conocimiento que se tiene del modelo DRBD y de herramientas de resolución se proponen distintos aspectos a considerar al momento de su aplicación. Estos quedan descritos en la Figura 5.1 que se aprecia a continuación:

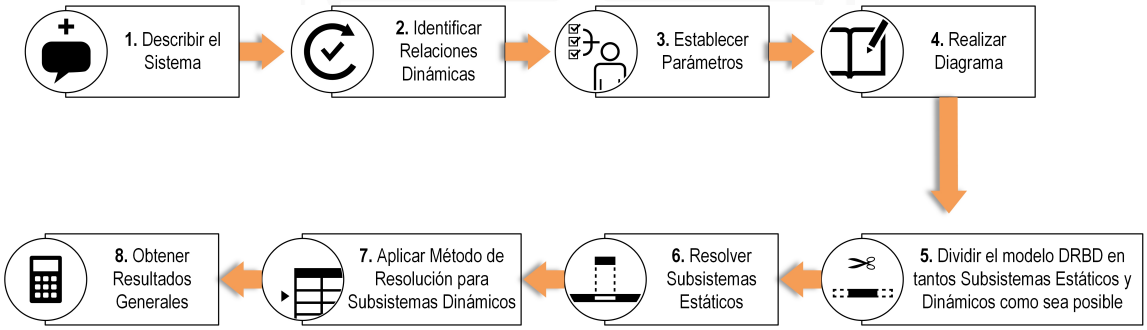


Figura 5.1: "Diagrama de propuesta metodológica"
(Fuente: Elaboración propia)

En lo que sigue se profundizará cada una de las etapas nombradas previamente.

5.1. Describir el sistema

En ésta primera instancia se debe conocer distintas partes del sistema a modelar, las que se describen a continuación:

Unidades. Consiste en una descripción individual de cada unidad perteneciente al sistema, especificando de qué equipo o máquina se trata, sus funciones, y sus valores nominales de operación.

Subsistemas. Consiste primero en un análisis de su existencia y luego en la descripción de éste, es decir, del grupo de unidades que trabajan de manera conjunta para responder

a las exigencias del proceso llevado a cabo. Es importante que si existen subsistemas, estos queden descritos a la vez en sus unidades más simples.

Procesos. Descripción de el/los proceso/s llevado/s a cabo en el sistema.

Luego, con la información detallada anteriormente será posible tener una noción clara del sistema con el que se está trabajando.

5.2. Identificar relaciones dinámicas

Como una barrera de entrada al modelamiento mediante DRBD es necesario determinar si existen propiedades dinámicas a ser tratadas, ya que, podría tratarse de un sistema que pudiera modelarse en forma más práctica mediante un modelamiento tradicional, RBD por ejemplo. De no realizarse este primer filtro se podría estar incurriendo en una actividad sin sustento, ya que no existiría beneficio por trabajar la confiabilidad del sistema de forma acuciosa.

Como se ha mencionado antes existen factores dinámicos a tener en consideración, estos son:

Variaciones con el tiempo. Se debe determinar si el estado de operación de una unidad evoluciona cuando una secuencia de eventos le ocurre.

Dependencias entre unidades. Se debe analizar si una unidad tiene incidencia (y/o es influenciada) en (por) el funcionamiento de otra unidad. Lo mismo se traduce al funcionamiento entre subsistemas o entre una unidad y un subsistema. Este aspecto es uno de mayor énfasis en el modelamiento DRBD, ya que, se establece la dependencia como la unidad base para construir el modelo de un sistema, es por ello que debe quedar claramente establecida cada dependencia existente.

Ambos factores descritos dan lugar a relaciones de dependencias que deben ser definidas, y que en su forma genérica fueron descritas en la Figura 4.6.

5.3. Establecer parámetros

Debido a que el modelamiento DRBD se centra en las componentes del sistema se deben establecer características con respecto a ellas. Y por otra parte dada las relaciones dinámicas identificadas se deben fijar los parámetros de la propia naturaleza del modelamiento DRBD. A continuación se describe cada uno de los elementos a definir:

Tasa de falla. Cada componente del sistema estará caracterizado por ésta medida que entregará información de cómo es recomendable modelarla en cuanto a confiabilidad.

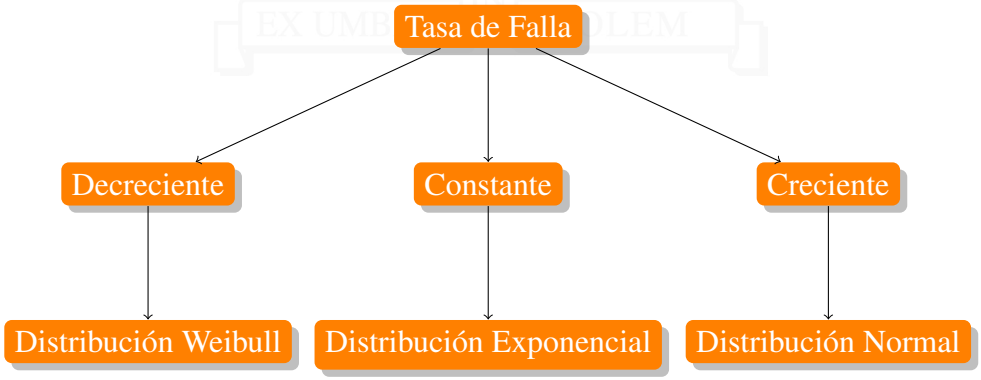


Figura 5.2: Árbol de decisión, tasa de falla.
(Fuente:Elaboración propia)

La Figura 5.2 muestra las distribuciones comúnmente utilizadas dependiendo del caso, sin embargo, como es sabido la distribución Weibull, dependiendo del valor de sus parámetros, puede modelar todos los estados en la vida de un elemento.

Tasa de reparación. Si el modelo considera la realización de reparaciones se debe integrar ésta característica para cada unidad.

Tasa de dependencia. Dado que se utilizará DRBD se debe fijar éste parámetro, es decir, el valor de β dependiendo de la relación de dependencia de la que se trate. Como se mencionó en el capítulo anterior, éste parámetro permite relacionar las tasas de falla de acuerdo a la Ecuación 4.8.

Prioridad. El elemento detotado por *c* del modelamiento DRBD, que indica cuando se debe activar una dependencia con respecto a otra(s) con la(s) que se encuentra en conflicto.

Probabilidad de propagación. En línea con el modelamiento DRBD se debe establecer el parámetro *p* que indica la probabilidad de que ocurra el evento reacción cuando ha ocurrido el evento trigger. Cuando no se indica se asume que es igual a 1.

5.4. Realizar diagrama

De acuerdo a la descripción del sistema, las relaciones dinámicas identificadas y los parámetros del modelo elegido, es recomendable la realización de un diagrama (bajo el modelamiento DRBD) que permita tener una representación más clara del sistema.

5.5. Dividir el modelo DRBD en tantos subsistemas estáticos y dinámicos como sea posible

Este paso toma en consideración el algorítmo de resolución del modelo DRBD visto previamente.

Se debe obtener cada subsistema de modo que sea independiente de los otros, es decir, que éste no comparte unidades con los otros subsistemas y que no existen dependencias entrantes o salientes hacia/desde él. Procediendo de ésta manera se puede acotar la resolución del sistema, dado que se aplica el mismo método de resolución a los susbsistemas estáticos que lo establecido por la teoría clasica de modelamiento RBD. Esto tiene sentido dado que un modelo DRBD sin parte dinámica no es más que un RBD.

Se debe considerar para discriminar entre un subsistema dinámico y uno estático la existencia o no de dependencias, siendo la existencia de dependencias la prueba de que el subsistema es dinámico. También servirá la identificación de propiedades dinámicas detallada en uno de los pasos previos.

5.6. Resolver subsistemas estáticos

Los subsistemas independientes estáticos identificados en el paso previo, como ya se ha señalado, pueden ser resueltos mediante la teoría clásica de confiabilidad aplicando las configuraciones del Diagrama de Bloques de Confiabilidad (RBD).

5.7. Aplicar método de resolución para subsistemas dinámicos

De acuerdo a lo señalada en la comparación de métodos de resolución, existen algunos que se recomiendan por encontrarse mejor evaluados en cuanto a nivel de complejidad, pero que no poseen tanta precisión dado los supuestos intrínsecos. Este es el caso de las CTMC y de las Petri Nets. Por otra parte, existen métodos que pueden alcanzar mayor grado de precisión, que es el caso de NMSPN y Simulación, y que sin embargo, poseen grados más altos de complejidad. La Figura 5.3 grafica lo señalado previamente.

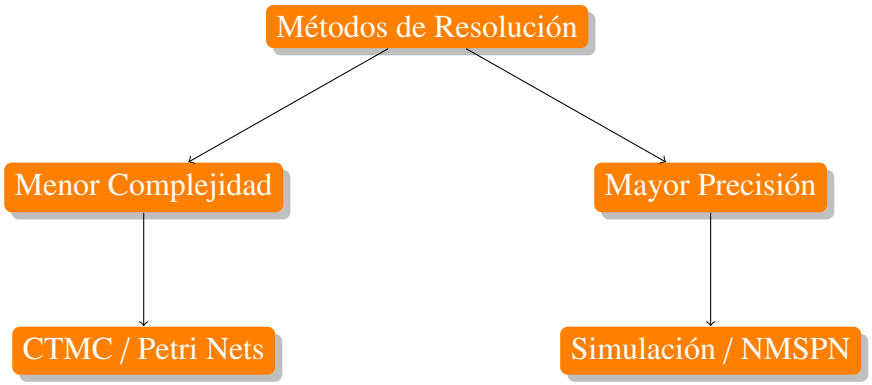


Figura 5.3: Diagrama métodos de resolución.
(Fuente:Elaboración propia)

Resumiendo, se deberá aplicar un método de acuerdo a la característica que se desee privilegiar.

Es importante destacar que el método de resolución elegido permite traducir el modelamiento DRBD realizado previamente a un dominio analizable para la obtención de resultados

5.8. Obtener resultados generales

En éste paso es necesario considerar las herramientas computacionales que pueden ser aplicadas dado un determinado método de resolución elegido para subsistemas dinámicos.

Una vez que se obtienen resultados para los subsistemas dinámicos es posible, de acuerdo al diagrama realizado para modelar el sistema, obtener la confiabilidad del sistema.

6 | CASO PRÁCTICO

El presente caso considerará el sistema de radar que fue modelado con Diagramas de Bloques de Confiabilidad en Klion (1992).

A continuación se aplicará la propuesta metodológica.

6.1. Describir el sistema

La función principal llevada a cabo por un sistema de radar es la medición de distancias, altitudes, direcciones y velocidades de objetos utilizando ondas electromagnéticas. Su funcionamiento se basa en emitir un impulso de radio, que se refleja en el objetivo y se recibe típicamente en la misma posición del emisor. El captar las ondas reflejadas permite obtener gran cantidad de información.

El sistema en estudio estará compuesto en forma general, por 5 subsistemas:

- Antena
- Receptor
- Transmisor
- Subsistema informático de seguimiento
- Controlador de radar

Con el objetivo de que el sistema de radar realice su función, la entrada debe ser correctamente procesada a través de la antena y el receptor/transmisor. Los computadores de seguimiento deben funcionar con los datos satisfactoriamente y el controlador del radar

debe adquirir una representación exacta de la información recibida para su transmisión a un sistema de comunicaciones. Si existe cualquier quiebre en el flujo de información, o si es introducida información o señales erróneas debido a una falla en cuaquiera de las componentes, entonces el sistema no puede seguir realizando su función y se considerará fallado.

El transmisor es la componente que emite el impulso y posteriormente el receptor es el que percibe la señal de vuelta, debido a esto una vez que las ondas han sido enviadas se apaga el transmisor y se enciende el receptor que espera el eco, y cuando nuevamente opere el transmisor se deberá tener apagado el receptor. Se debe considerar que el transmisor está compuesto por una fuente de poder, un modulador, y un amplificador.

El sistema informático de seguimiento posee dos computadoras separadas para realizar la función de seguimiento. “Dos son necesarias para manejar la carga peak, pero una computadora puede realizar la función adecuadamente pero menos eficientemente si es requerido.” (Klion, 1992)

6.2. Identificar relaciones dinámicas

Se visualiza una dependencia entre ambas computadoras, ya que, como se señaló anteriormente, una de las dos computadoras funcionando permite que el sistema siga operando, aun cuando la eficiencia de la componente restante sea menor. Este comportamiento menos eficiente sugiere que el equipo se sobrecarga, lo que enmascara un incremento en su tasa de falla al encontrarse soportando la carga del sistema.

La conducta descrita para los computadores de seguimiento puede modelarse como load sharing debido a que la falla de una computadora provoca una sobrecarga en la restante, lo que haría incrementar su tasa de falla.

Es posible apreciar una dependencia entre el receptor y el transmisor, dado que cuando uno de ellos se encuentra operando el otro no estará en operación. Este comportamiento será modelado como una dependencia Sleep/Wake-up, ya que, una vez que el transmisor deja de prestar función, se comienza con la recepción y éste último equipo debe activarse.

6.3. Establecer parámetros

Tasa de falla. Para continuar con el ejemplo establecido en Klion (1992), se modelará la tasa de falla como constante, y por ello será posible utilizar una distribución exponencial para modelar la confiabilidad de las componentes. En la Tabla 6.1 se presentan las tasas de falla de las componentes:

Tabla 6.1: Tasas de falla de un sistema de radar.

Componente	**Notación**	**Tasa de falla** $[h^{-1}]$
Antena	λ_A	0,000125
Receptor	λ_R	0,0009
Fuente de poder	λ_F	0,0005
Modulador	λ_M	0,0002
Amplificador	λ_{AM}	0,0002
Computadora de seguimiento	λ_C	0,0002
Controlador	λ_{CO}	0,00015

Tasas de Reparación. El modelo no considerará reparación.

Tasas de dependencia. Para la relación de load sharing se considerará $\beta = 1,1$ de modo que se refleje el efecto negativo en la tasa de falla de la componente restante una vez que una falla. Para la dependencia Sleep/Wake-up se establecerá una tasa de dependencia de 1, lo que reflejaría que la componente en dependiente se encuentra totalmente energizada para hacer el cambio en la función del sistema de radar.

Prioridades. Dado que las relaciones de dependencia se encuentran en subsistemas separados no se vislumbran conflictos entre ellas al momento de ser aplicadas.

Probabilidades de propagación. Para ambas dependencia se fijará una probabilidad de 100 % ($p = 1$).

6.4. Realizar diagrama

En la Figura 6.1 se encuentra el diagrama realizado para el sistema de radar utilizando los parámetros y relaciones descritas con anterioridad.

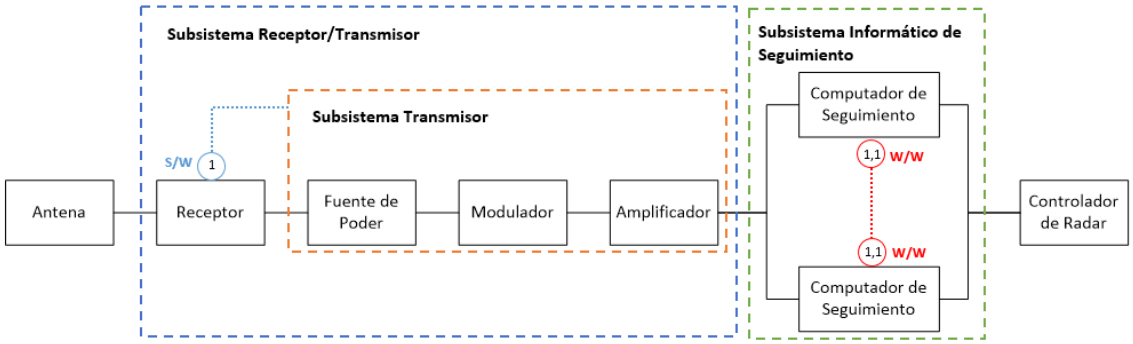


Figura 6.1: "DRBD para sistema de radar"

(Fuente: Elaboración Propia)

Al mismo tiempo la Figura 6.1 se puede comparar con el diagrama encontrado en la literatura para el sistema de radar modelado con RBD se puede apreciar en la Figura 6.2.

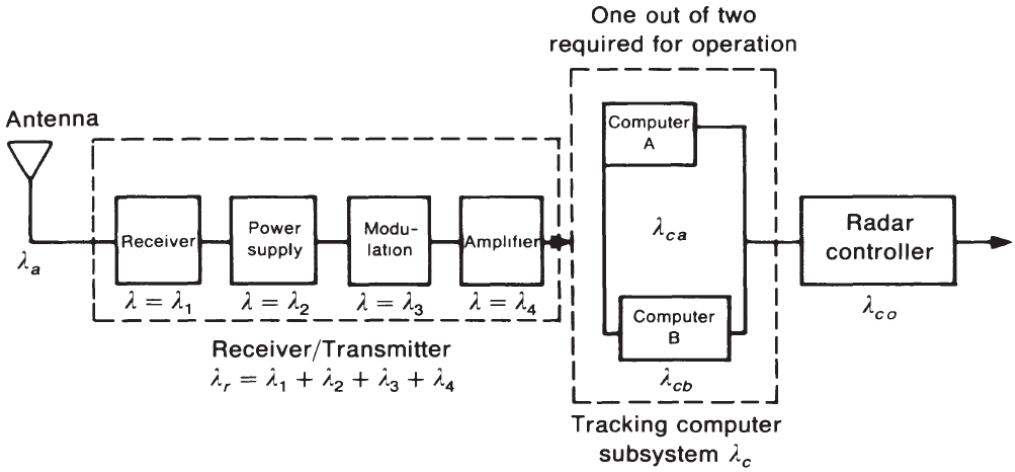


Figura 6.2: "Diagrama de bloques de confiabilidad para sistema de radar"

(Fuente: Practical electronic reliability engineering)

6.5. Dividir el modelo DRBD en tantos subsistemas estáticos y dinámicos como sea posible

En la Figura 6.1 es posible notar los siguientes subsistemas:

- **Subsistema Transmisor.** Es un sistema estático que se compone de la fuente de poder, el modulador y el amplificador. Como se requiere de los tres elementos para funcionar su configuración es serie.
- **Subsistema Receptor/Transmisor.** Es un sistema dinámico que se encuentra en dependencia Sleep/Wake-up.
- **Subsistema informático de seguimiento.** Es un sistema dinámico que considera una relación load sharing entre las computadoras de seguimiento.

6.6. Resolver subsistemas estáticos

De acuerdo a lo señalado en el punto anterior se tiene un subsistema estático que corresponde al sistema transmisor con una configuración de serie. La confiabilidad de éste subsistema corresponde a lo siguiente:

$$R_T = e^{-\lambda_F t} \cdot e^{-\lambda_M t} \cdot e^{-\lambda_{AM} t} = e^{-\lambda_T t} \tag{6.1}$$

Además se tiene que $\lambda_T = \lambda_F + \lambda_M + \lambda_{AM}$

6.7. Aplicar método de resolución para subsistemas dinámicos

Se utilizará para resolver las partes dinámicas encontradas, el método de Continuous Time Markov Chains (CTMC) dado que se privilegiará la menor complejidad en la obtención de resultados.

Subsistema informático de seguimiento

Para resolver el sistema informático de seguimiento que se encuentra en load sharing primero que nada se establecerán los estados del sistema. Como en este análisis el estado stand by no juega un rol preponderante, se considerarán solo dos estados para cada componente, a saber, operacional (O), y fallado (F). Se obtiene la siguiente tabla:

Tabla 6.2: Tabla de estados sistema informático de seguimiento.

Computadora 1	Computadora 2	Número de estado	Utilidad
O	O	1	✓
O	F	2	✓
F	O	3	✓
F	F	4	×

Dado que como las computadoras son idénticas, se asume que también lo son sus tasas de falla independientes, en consecuencia el número de estados se puede reducir como se muestra en la Tabla 6.3.

Tabla 6.3: Tabla simplificada de estados, sistema informático de seguimiento.

Computadora 1	Computadora 2	Número de estado	Utilidad
O	O	1	✓
O	F	2	✓
F	O	2	✓
F	F	3	×

Una vez conocidos los estados del sistema se procederá a realizar el diagrama de transición de estados que se presenta en la siguiente figura:

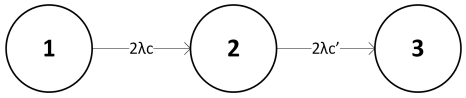


Figura 6.3: "Diagrama de transición de estados, subsistema en informático de seguimiento"
(Fuente: Elaboración propia)

Si bien es cierto la tasa de falla de las computadoras idénticas es conocida, para obtener las tasas de fallas posterior a la falla de una de las componentes se utilizará la Ecuación 4.8 lo que deja al descubierto la aplicación práctica de lo que se conoce teóricamente como el aumento de la tasa de falla en una componente restante cuando una de las componentes falla al encontrarse estas en configuración load sharing.

Dado el diagrama de transición de estados es posible realizar la matriz de transición de estados correspondiente:

$$M_{SIS} = \begin{bmatrix} -2\lambda_C & 0 & 0 \\ 2\lambda_C & -2\lambda_C^{'} & 0 \\ 0 & 2\lambda_C^{'} & 0 \end{bmatrix} \tag{6.2}$$

Luego, se obtiene el siguiente sistema de ecuaciones:

$$(1) \qquad P_1(t)' = -2\lambda_C \cdot P_1(t) \tag{6.3}$$

$$(2) \qquad P_2(t)' = 2\lambda_C \cdot P_1(t) - 2\lambda_C^{'} \cdot P_2(t) \tag{6.4}$$

$$(3) \qquad P_3(t)' = 2\lambda_C^{'} \cdot P_2(t) \tag{6.5}$$

Resolviendo el sistema de ecuaciones anterior se obtienen las probabilidades de cada estado en el tiempo t:

$$P_1(t) = e^{-0,00084t} \cdot (-5,14063 \cdot 10^{-17} \cdot e^{0,0004t} + e^{0,00044t}) \tag{6.6}$$

$$P_2(t) = -10 \cdot e^{-0,00084t} \cdot (e^{0,0004t} - e^{0,00044t}) \tag{6.7}$$

$$P_3(t) = e^{-0,00084t} \cdot (10 \cdot e^{0,0004t} - 11 \cdot e^{0,00044t} + e^{0,00084t}) \tag{6.8}$$

Finalmente la confiabilidad del sistema estará dada de acuerdo a lo siguiente:

$$R_{SIS}(t) = P_1(t) + P_2(t) \tag{6.9}$$

$$R_{SIS}(t) = -10 \cdot e^{-0,00044t} + 11 \cdot e^{-0,0004t} \tag{6.10}$$

Subsistema receptor/transmisor

También se utiliza como método de resolución CTMC, y para éste subsistema en particular es relevante considerar el estado stand by, por ello se considerarán tres estados en los que se puede encontrar una componente: activo (A), stand by (S), y fallado (F). A continuación, la Tabla 6.4 muestra los diferentes estados del subsistema.

Tabla 6.4: Tabla de estados, subsistema receptor/transmisor.

Receptor	Transmisor	Número de estado	Utilidad
A	A	1	×
A	S	2	✓
A	F	3	×
S	A	4	✓
S	S	5	×
S	F	6	×
F	A	7	×
F	S	8	×
F	F	9	×

Aunque explícitamente no está modelado en el diagrama DRBD, las componentes se encuentran en serie dado los requerimientos del sistema. Sin embargo, el hecho que no sea útil que cualquiera de las dos componentes se encuentre fallada si se puede integrar en el método de resolución CTMC posteriormente al considerar solo los estados útiles.

Es importante destacar que como la tasa de dependencia elegida en la dependencia del subsistema es igual a 1, una componente en estado stand by tiene la misma tasa de falla que si estuviera activa, dado que se encuentra completamente energizada.

Como ya fue mencionado el estado stand by es relevante, por ello para utilizar el método de resolución es necesario además fijar las probabilidades de transición de ambas componentes del subsistema tanto de pasar de un estado activo a uno stand by como de pasar del estado stand by al estado activo. Para ésto último se considerará el tiempo medio entre transmisiones (MTBTr) y el tiempo medio entre recepciones (MTBRec) como se muestra en la Figura 6.4.

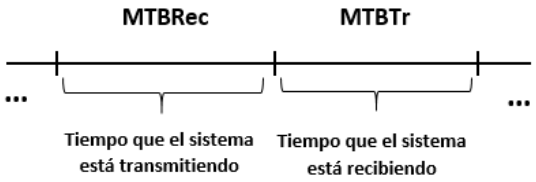


Figura 6.4: "Tiempos medios de transmisión y recepción"
(Fuente: Elaboración propia)

En resumen se tienen las siguientes probabilidades de transición:

Tabla 6.5: Tabla probabilidades de transición.

Probabilidad	Receptor	Transmisor
$P_{A\longrightarrow S}$	p_{R1}	p_{T1}
$P_{A\longrightarrow F}$	λ_R	λ_T
$P_{S\longrightarrow F}$	λ_R	λ_T
$P_{S\longrightarrow A}$	p_{R2}	p_{T2}
$P_{F\longrightarrow A}$	0	0
$P_{F\longrightarrow S}$	0	0

Para éste caso se considerará un tiempo promedio de transmisión de 5 horas y un tiempo promedio de recepción de 5 horas. Entonces para el transmisor se tiene:

$$P_{A\longrightarrow S} = \frac{1}{MTBRec} = 0,2 = p_{T1} \tag{6.11}$$

$$P_{S\longrightarrow A} = \frac{1}{MTBTr} = 0,2 = p_{T2} \tag{6.12}$$

Y para el receptor se tiene:

$$P_{A\longrightarrow S} = \frac{1}{MTBTr} = 0,2 = p_{R1} \tag{6.13}$$

$$P_{S\longrightarrow A} = \frac{1}{MTBRec} = 0,2 = p_{R2} \tag{6.14}$$

La información anterior se puede resumir en el diagrama de transición de estados de la Figura 6.5.

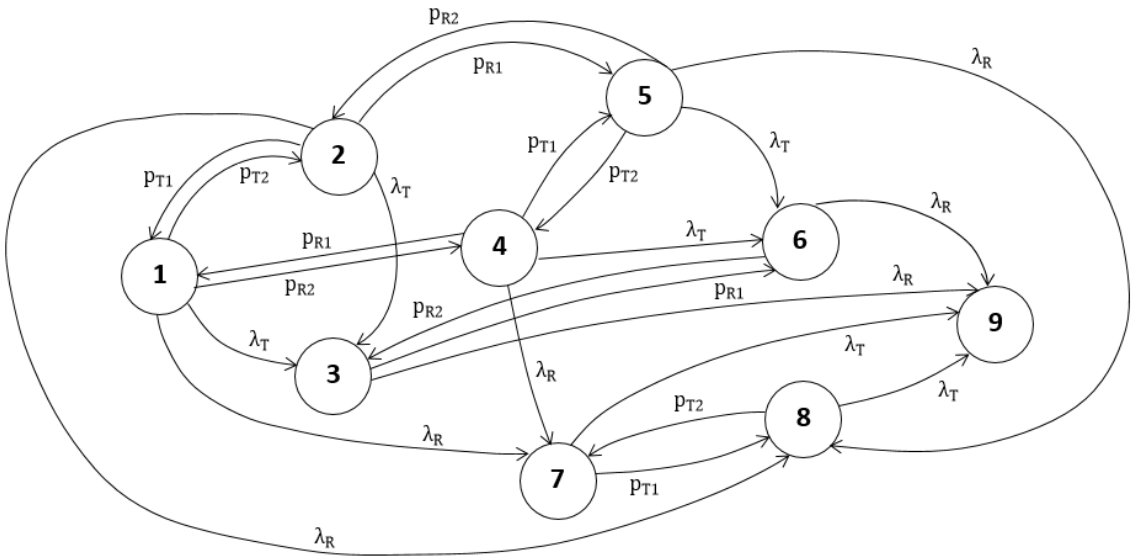


Figura 6.5: "Diagrama de transición de estados, subsistema receptor/transmisor"
(Fuente: Elaboración propia)

A partir del diagrama de la Figura 6.5 es posible obtener la matriz de transición del subsistema en cuestión:

$$M_{RT} = \begin{pmatrix} (-p_{T1}-p_{R1}-\lambda_T-\lambda_R) & p_{T2} & 0 & p_{R2} & 0 & 0 & 0 & 0 & 0 \\ p_{T1} & (-p_{R1}-p_{T2}-\lambda_R) & 0 & 0 & p_{R2} & 0 & 0 & 0 & 0 \\ \lambda_T & \lambda_T & (-p_{R1}-\lambda_R) & 0 & 0 & p_{R2} & 0 & 0 & 0 \\ p_{R1} & 0 & 0 & (-\lambda_T-p_{T1}-p_{R2}-\lambda_R) & p_{T2} & 0 & 0 & 0 & 0 \\ 0 & p_{R1} & 0 & p_{T1} & (-p_{T2}-\lambda_T-p_{R2}-\lambda_R) & 0 & 0 & 0 & 0 \\ 0 & 0 & p_{R1} & \lambda_T & \lambda_T & (-\lambda_R-p_{R2}) & 0 & 0 & 0 \\ \lambda_R & 0 & 0 & \lambda_R & 0 & 0 & (-\lambda_T-p_{T1}) & p_{T2} & 0 \\ 0 & \lambda_R & 0 & 0 & \lambda_R & 0 & p_{T1} & (-p_{T2}-\lambda_T) & 0 \\ 0 & 0 & \lambda_R & 0 & 0 & \lambda_R & \lambda_T & \lambda_T & 0 \end{pmatrix}$$

Figura 6.6: "Matriz de transición, subsistema receptor/transmisor"
(Fuente: Elaboración propia)

A partir de la matriz de transiciones se puede representar el sistema de 9 ecuaciones diferenciales de la siguiente forma:

$$\frac{d}{dt}P(t) = M_{RT} \times P(t) \tag{6.15}$$

Donde $P(t)$ representa el vector con las probabilidades de cada estado en el tiempo t.

Resolviendo el sistema de ecuaciones diferenciales con la ayuda de un sooftware se obtienen los resultados de probabilidades para cada uno de los estados, pero en términos de la confiabilidad del subsistema serán de interés los resultados de los estados 2 y 4:

$$P_2(t) = 0,249298 \cdot e^{-2,4099t} \cdot (e^{1,60832t} - 4,14532 \cdot 10^{-26} \cdot e^{2,0081t} + 2,00563 \cdot e^{2,00855t} + 7,24812 \cdot 10^{-13} \cdot e^{2,009t} + 1,00564 \cdot e^{2,40833t} + 1,73861 \cdot 10^{-14} \cdot e^{2,409t}) \tag{6.16}$$

$$P_4(t) = 0,249859 \cdot e^{-2,4099t} \cdot (e^{1,60832t} + 4,17678 \cdot 10^{-26} \cdot e^{2,0081t} - 2,00113 \cdot e^{2,00855t} - 7,23201 \cdot 10^{-13} \cdot e^{2,009t} + 1,00113 \cdot e^{2,40833t} + 1,81411 \cdot 10^{-14} \cdot e^{2,409t}) \tag{6.17}$$

La confiabilidad del subsistema queda de la siguiente forma:

$$R_{RT}(t) = P_2(t) + P_4(t) \tag{6.18}$$

$$R_{RT}(t) = 0,499157 \cdot e^{-0,801575t} + 1,01851 \cdot 10^{-28} \cdot e^{-0,4018t} - 1,26562 \cdot 10^{-6} \cdot e^{-0,40135t} - 4,21625 \cdot 10^{-18} \cdot e^{-0,4009t} + 0,500844 \cdot e^{-0,00157468t} + 8,86703 \cdot 10^{-15} \cdot e^{-0,0009t} \tag{6.19}$$

6.8. Obtener resultados generales

La obtención de la confiabilidad del sistema general de radar consiste en la consideración en serie (estática) de la antena, el subsistema Receptor/Transmisor, el subsistema informático de seguimiento, y el controlador del radar. Lo anterior se puede ver representado de la siguiente forma:

$$R_{SR}(t) = R_A(t) \cdot R_{RT}(t) \cdot R_{SIS}(t) \cdot R_{CO}(t) \tag{6.20}$$

Resumidamente se tienen las siguientes funciones de confiabilidad para cada subsistema mencionado:

$$R_A = e^{-0,000125t} \tag{6.21}$$

$$R_{RT}(t) = 0,499157 \cdot e^{-0,801575t} + 1,01851 \cdot 10^{-28} \cdot e^{-0,4018t} - 1,26562 \cdot 10^{-6} \cdot e^{-0,40135t} \\ -4,21625 \cdot 10^{-18} \cdot e^{-0,4009t} + 0,500844 \cdot e^{-0,00157468t} + 8,86703 \cdot 10^{-15} \cdot e^{-0,0009t} \tag{6.22}$$

$$R_{SIS}(t) = -10 \cdot e^{-0,00044t} + 11 \cdot e^{-0,0004t} \tag{6.23}$$

$$R_{CO}(t) = e^{-0,00015t} \tag{6.24}$$

Con la función de confiabilidad dinámica descrita previamente y la función de confiabilidad estática, reconstruida como un sistema en serie de todas las componentes según se aprecia de la Figura 6.2, es posible graficar el comportamiento de la confiabilidad en el tiempo para el sistema de radar como se muestra en la Figura 6.7:

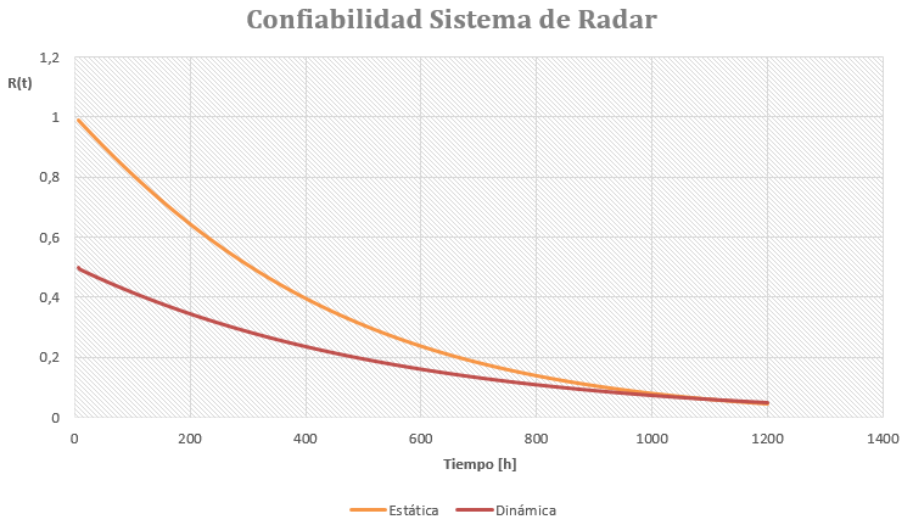


Figura 6.7: "Confiabilidad del sistema de radar"
(Fuente: Elaboración propia)

Como se puede apreciar en la Figura 6.7 la confiabilidad dinámica modelada para el sistema en el tiempo es menor a la confiabilidad estática hasta un punto en donde ambas aproximadamente convergen. Esto se debe a los supuestos establecidos para el sistema en función de acercarlo más a su propia realidad, que tiene comportamientos dinámicos.

7 | CONCLUSIONES

En el presente trabajo se realizó una revisión general de la teoría clásica de confiabilidad en donde se pudo apreciar que los elementos considerados en el modelamiento de la confiabilidad de sistemas son: el sistema como un todo (definición), los elementos que lo componen, sus cantidades, sus cualidades, y el funcionamiento del sistema que en forma más práctica corresponde al arreglo entre las componentes.

De acuerdo al estudio realizado de modelamiento de sistemas dinámicos sus propiedades básicas corresponden a la *variación en el tiempo* del sistema, que se da por el cambio de estado de sus componentes en el tiempo; y la *dependencia entre componentes*, que se crea por medio de la asociación de eventos a las relaciones entre componentes.

El estudio del modelamiento dinámico de confiabilidad nace por la necesidad del área de la informática (Computer science) de integrar a la evaluación de confiabilidad y otras medidas, los comportamientos reales en el funcionamiento de los sistemas de investigación propia de su campo, por ejemplo sistemas computacionales distribuidos. Esto con el objetivo de obtener resultados más precisos, ya que, como es muy comentado en la literatura asociada, en éste tipo de sistemas se tiende a sobrevalorar la confiabilidad cuando se utilizan herramientas estáticas.

En éste trabajo fue posible saber sobre la necesidad cada vez más creciente de contar con modelos dinámicos de confiabilidad y conocer formas de modelamiento dinámico principalmente a lo largo del estudio del modelo DRBD, pero también se pudo conocer el modelo DFT y la otra línea de investigación del modelo DRBD aunque en menor profundidad.

El modelamiento de confiabilidad dinámica DRBD ayuda gráficamente a entender el sistema en cuestión, agregando más datos con respecto a dinámica y utilizando como base

uno de los modelos más conocidos y utilizados en modelamiento de confiabilidad (RBD), pero la resolución final es algo más compleja, ya que, es necesario conocer métodos de resolución acogiéndose a los requerimientos propios de estos últimos.

Se establecieron los pasos para aplicar el modelo DRBD en la "Propuesta metodológica" donde se considera: (1) Describir el sistema, (2) Identificar relaciones dinámicas, (3) Establecer parámetros, (4) Realizar diagrama, (5) Dividir el modelo DRBD en tantos subsistemas estáticos y dinámicos como sea posible, (6) Resolver subsistemas estáticos, (7) Aplicar el método de resolución para subsistemas dinámicos, y (8) Obtener resultados generales.

En el "Caso práctico" se elaboró un modelamiento dinámico de confiabilidad utilizando el modelo DRBD. El método de resolución utilizado en éste (CTMC) permite considerar en forma más precisa los distintos estados útiles de la relación de dependencia entre componentes utilizada, en función de obtener la confiabilidad del sistema. Es importante destacar que la resolución comienza a complejizarse a medida que aumenta el número de estados (state space explotion) dado que se debe resolver un sistema de ecuaciones diferenciales, y para ésto último es casi imprescindible contar con la ayuda de un software.

Los resultados obtenidos para un modelamiento dinámico de un sistema dependerán de los supuestos realizados en base a su información específica de funcionamiento, sin embargo, de acuerdo a lo visto en la literatura relacionada y en el caso práctico del presente trabajo, en general se tendrá una menor confiabilidad en comparación con el mismo sistema modelado en forma estática.

Finalmente es importante mencionar que este trabajo considera una limitante, especialmente del caso práctico, que corresponde a la no inclusión del mantenimiento en el análisis. Esto con la intención de hacer más simple el estudio y resolución de lo ejemplificado.

Bibliografía

Arata (2009). *Ingeniería y gestión de la confiabilidad operacional en plantas industriales. Aplicación de la plataforma R-MES*. 4.1.1, 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.1.5

Barabadi; Barabady; y Markeset (2011). A methodology for throughput capacity analysis of a production facility considering environment condition. *Reliability Engineering and System Safety*, 96(12), 1637–1646. 2

Bobbio, Andrea; Puliafito, Antonio; y Tekel, Miklós (2000). A Modeling Framework to Implement Preemption Policies in Non-Markovian SPNs. 26(1), 36–54. 4.6.2

Bolch, Gunter; Greiner, Stefan; de Meer, Hermann; y Trivedi, Kishor (2006). *Queueing Networks and Markov Chains*. 4.6.2, 4.6.2

Bourouni (2013). Availability assessment of a reverse osmosis plant: Comparison between Reliability Block Diagram and Fault Tree Analysis Methods. *Desalination*, 313, 66–76. 2

Coit, David W.; Chatwattanasiri, Nida; Wattanapongsakorn, Naruemon; y Konak, Abdullah (2015). Dynamic k-out-of-n system reliability with component partnership. *Reliability Engineering & System Safety*, 138, 82–92. 2

Crespo, A. (2014). *The Maintenance Management Framework*. Número 1. 4.1.2

Distefano y Puliafito (2006). System modeling with dynamic reliability block diagrams. *Safety and Reliability for Managing Risk*, (pp. 141–150). 2, 4.1.5, 4.2, 4.2.1, 4.2.2, 4.2.2, 4.2.3, 4.2.3, 4.2.4, 4.2.4

Distefano y Puliafito (2009). Reliability and availability analysis of dependent–dynamic systems with DRBDs. *Reliability Engineering & System Safety*, 94(9), 1381–1393. 2, 4.2.5, 4.2.6, 4.6.1, 4.6.4

Distefano; Scarpa; y Puliafito (2006). Modeling distributed computing system reliability with DRBD. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, (pp. 106–115). 2, 4.2, 4.2.1, 4.2.1, 4.2.2, 4.2.2, 4.2.2, 4.2.3, 4.5

Distefano, Salvatore y Puliafito, Antonio (2007). Dynamic Reliability Block Diagrams VS Dynamic Fault Trees. *IEEE*, (pp. 71–76). 4.3, 4.4

Distefano, Salvatore y Xing, Liudong (2006). A New Approach to Modeling the System Reliability : Dynamic Reliability Block Diagrams. 00(C), 189–195. 4.5, 4.5

Dubrova, Elena (2013). *Fault-Tolerant Design*. 4.6.2

Ebrahimi (2010). Effect analysis of Reliability, Availability, Maintainability and Safety (RAMS) Parameters in design and operation of Dynamic Positioning (DP) systems in floating offshore structures. *Master Thesis written at KTH, Royal Institute of Technology, School of Industrial Engineering*, (October). 4.1.4

Estay (2014). Comparación de teorías clásicas de confiabilidad con el uso de Cadenas de Markov y Redes de Petri. *Thesis written at Universidad Técnica Federico Santa María.* 4.1.5

Høyland y Rausand (1994). *System reliability theory: models and statistical methods*. 4.1.2

Hrúz, Branislav y Zhou, MengChu (2007). *Modelling and control of discrete-event dynamic systems*. 4.6.2

Kim (2011). Reliability block diagram with general gates and its application to system reliability analysis. *Annals of Nuclear Energy*, 38(11), 2456–2461. 2

Klion, Jerome (1992). *Practical Electronic Reliability Engineering*. 6, 6.1, 6.3

Liudong, Xing y Xu, Haiping (2007). Formal Semantics and Verification of DRBD for system reliability modeling. (September), 1–8. 4.5

Melo; Lara; y Jacobo (2009). Estimación de la confiabilidad-disponibilidad- mantenibilidad mediante una simulación tipo Monte Carlo de un sistema de compresión de gas amargo durante la etapa de ingeniería. *Tecnol. Ciencia Ed (IMIQ)*, 24(2), 93–104. 4.1.1, 4.1.4, 4.6.2

Rogge-solti, Andreas y Weske, Mathias (2015). Prediction of business process durations using non-Markovian stochastic Petri nets. *Information Systems*, 54, 1–14. 4.6.2

Sutton, Ian (2010). *Process Risk and Reliability Management. Operational Integrity Management*. 4.6.2, 4.6.2

Volovoi, V (2004). Modeling of system reliability Petri nets with aging tokens. 84, 149–161. 4.6.2

Xu; Xing; y Robidoux (2008). DRBD - Dynamic Reliability Block Diagrams for System Reliability Modelling. *International journal of computers applications*, 31(2), 132–141. 2, 4.2, 4.5, 4.5, 4.6.4