Razonamiento Temporal Aplicado al Monitoreo con Respecto al Tiempo en el Desarrollo de Actividades de un Proyecto

T E S I S

Maestría en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Ing. Sara Elena Garza Villarreal

Diciembre 2006

Razonamiento Temporal Aplicado al Monitoreo con Respecto al Tiempo en el Desarrollo de Actividades de un Proyecto

TESIS

Maestría en Ciencias en

Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Ing. Sara Elena Garza Villarreal

Diciembre 2006

Instituto Tecnológico y de Estudios Superiores de Monterrey

División de Graduados en Tecnologías de Información y Electrónica

Dirección de Programas de Posgrado en Tecnologías de Información y Electrónica

Los miembros del comité de tesis recomendamos que la presente tesis de Sara Elena Garza Villarreal sea aceptada como requisito parcial para obtener el grado académico de Maestro en Ciencias en:

Sistemas Inteligentes

Comité de tesis:

Dr. José Luis Aguirre Cervantes
Asesor de la tesis

Dr. Ramón F. Brena Pinero
Sinodal

Dr. Miguel Ángel Pérez Guardado
Sinodal

Graciano Dieck Assad, PhD.
Director del Programa de Graduados en
Tecnologías de Información y Electrónica

Diciembre de 2006

Razonamiento Temporal Aplicado al Monitoreo con Respecto al Tiempo en el Desarrollo de Actividades de un Proyecto

Por

Ing. Sara Elena Garza Villarreal

TESIS

Presentada a la División de Tecnologías de Información y Electrónica
Este trabajo es requisito parcial para obtener el grado académico de Maestro en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Monterrey, N.L. Diciembre de 2006

A mis abuelos: Sara, Antonio, Rebeca (+), Angelmira y Félix . . . y a Karol.

Reconocimientos

Deseo externar un sincero agradecimiento a las personas que de alguna forma colaboraron en el desarrollo de esta tesis.

A Dios, por darme la vida y todo cuanto poseo. Sin Él, nada de esto sería posible ...

A mi familia por apoyarme siempre en todo lo que hago. Este logro es también de ustedes.

Al Dr. José Luis Aguirre por asesorar esta tesis y en general por todo el apoyo que me ha brindado desde que estaba en la carrera profesional. Se lo agradezco de sobremanera ...

Al Dr. Ramón F. Brena, por contribuir a la realización de esta tesis y por el apoyo brindado a través de la Cátedra de Tecnologías de Conocimiento y Agentes Inteligentes. Asimismo, gracias al Dr. Miguel Ángel Pérez por cedernos parte de su tiempo y ayudarnos al formar parte del comité.

Al Dr. Hugo Terashima también por su ayuda y dirección, en especial este último semestre.

A mis compañeros del CSI, sobretodo a Eduardo H. Ramírez, Jesús Héctor Domínguez, Claudia Farías y César Marín por el apoyo y orientación que me dieron durante este trayecto.

A todos mis amigos, en especial a Lorena y Adriana, porque les tocó estar un poco más de cerca con este trabajo. Muchas gracias por sus ánimos y su ayuda.

A todas las personas que estuvieron involucradas de alguna manera en la realización de esta tesis ...

Sara Elena Garza Villarreal

Instituto Tecnológico y de Estudios Superiores de Monterrey
Diciembre 2006

Razonamiento Temporal Aplicado al Monitoreo con Respecto al Tiempo en el Desarrollo de Actividades de un Proyecto

Sara Elena Garza Villarreal, M.C.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2006

Asesor de la tesis: Dr. José Luis Aguirre Cervantes

El presente trabajo trata acerca de plantear un mecanismo de monitoreo que sea capaz de detectar discrepancias de origen temporal (atrasos, adelantos) en las actividades pertenecientes a un proyecto. Debido a que cada vez es más común ver en el entorno organizacional que algunas de las estrategias se enfocan en el desarrollo de proyectos, éstos se vuelven una pieza relevante para las organizaciones. Por tanto, para que cumplan bien con su función, es necesario que sean administrados; si bien la administración de proyectos toma en cuenta varios rubros—tales como presupuesto, alcance y calidad—, se puede decir que uno de los más importantes es el del *tiempo*. Aunada a esto, se encuentra la cuestión del creciente uso de las tecnologías de información y conocimiento para contribuir al desarrollo de los proyectos. Por tanto, se vuelve deseable explorar alternativas que en cierto sentido pudieran favorecer la realización automatizada de procesos involucrados en un proyecto. En consecuencia, lo que se propone en la presente investigación es utilizar el Razonamiento Temporal (en su representación de álgebra de intervalos) para construir un mecanismo que monitoree la dimensión temporal del proyecto. Para ello, se plantea ver en una red de trabajo a las actividades como intervalos y a las relaciones de precedencia entre éstas como hechos que sirven para derivar nuevas relaciones, las cuales a su vez constituyen un esquema teórico que debe darse en el desarrollo del proyecto para evitar discrepancias temporales. El mecanismo construido bajo esta conceptualización fue probado y se pudo observar que sí cumple el propósito de identificar dichas discrepancias en la mayoría de los casos.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

Cada vez, en el ámbito organizacional, es más común encontrar estrategias basadas en el desarrollo de proyectos. Debido a la relevancia de éstos, cabe destacar que para que cumplan con su función, es necesario que sean *administrados*. De hecho, "la gerencia del proyecto es una competencia crítica en las organizaciones" [5].

Siguiendo esta misma línea, dentro de la administracion de proyectos, existen detalles importantes que deben considerarse para procurar que éstos sean exitosos. Por ejemplo, el alcance, el presupuesto, los recursos (que por lo general son limitados), los riesgos existentes a lo largo de su duración y el *tiempo* necesario para llevar éstos a cabo. De los anteriores, este último rubro es de gran interés debido a diversos motivos. En primer lugar, el tiempo implica costos; si el producto/servicio no se completa dentro del tiempo estipulado para ello, muy probablemente habrá que invertir más recursos (humanos, monetarios, etc.) para sacar el proyecto adelante, causando con esto que se rebase el presupuesto inicialmente asignado, aumenten los costos y se incrementen los riesgos. Asimismo, dada la importancia que cobra la competencia, cuando no se actúa con suficiente velocidad, se corre el peligro de salir al mercado cuando ya hay varios que se han adelantado en el camino. Un riesgo todavía de mayor impacto es que el proyecto se vuelva obsoleto de no estar listo a tiempo. Por ejemplo, considérese un sistema computacional de inscripciones; si éste no se encuentra implantado y en estado funcional para el día en que se ocupa, de nada habrá servido todo el esfuerzo y recursos invertidos. El consumidor de este producto no puede darse el lujo de esperar hasta que el sistema se encuentre preparado, por lo que el proyecto habrá fracasado. La empresa responsable de proveer este producto posiblemente perderá al cliente, reputación y futuros negocios debido a que no supo manejar un recurso muy importante, que *permite* o *impide* al proyecto llegar a un feliz término. Es por ello que la *dimensión temporal* de un proyecto es uno de los pilares que determina su éxito o fracaso.

Asimismo, con el creciente uso de las tecnologías de información como herramientas para contribuir a su realización, resulta frecuente encontrar dentro del desarrollo de los proyectos entidades que se encargan de ejecutar tareas de forma automática, con poca intervención humana. Ante esto, se vuelve deseable el explorar alternativas que en cierto sentido pudieran favorecer la realización automatizada de procesos involucrados en el proyecto [18].

Todo lo anterior—la importancia de la administración de un proyecto, en especial con respecto al tiempo y la necesidad de explorar alternativas que involucren el uso de tecnologías de información en un proyecto—es parte de la motivación para la presente tesis.

1.1. Definición del problema

Para controlar el tiempo dentro de un proyecto de tal suerte que éste se consuma a como es debido, se requiere llevar a cabo—aunado al proceso de control—un proceso de seguimiento o "monitoreo" para verificar que se esté trabajando correctamente. No obstante, la función de monitoreo plantea varias interrogantes a ser resueltas:

- ¿En qué consiste el proceso de seguimiento (monitoreo)?
- ¿Qué elementos se consideran para realizar el monitoreo?
- ¿Cómo se ha de ejecutar el seguimiento?
- ¿Qué criterios se utilizan para definir una situación "normal" de una que no lo es?
- ¿Cuándo y con qué frecuencia es necesario hacerlo?
- ¿Hay excepciones y/o casos especiales? ¿Cómo se manejan?

Estas preguntas definen de manera general el problema: *realizar el proceso de monitoreo del tiempo en las actividades de un proyecto*. De hecho, cada una de ellas se puede ver como un subproblema.

Ahora bien, considerando que estas interrogantes se pueden manejar como "variables", es posible fijarlas para explorar uno sólo de los subproblemas que contiene el monitoreo temporal en un proyecto. De esta forma: 1) se puede considerar al proceso de monitoreo como aquél en el cual se revisan las actividades para compararlas con respecto al plan de trabajo; 2) con respecto a los elementos, éstos consisten en las diversas actividades que se llevan a cabo dentro del proyecto (de hecho, algunas de éstas son críticas, ya que si una se retrasa, todo el proyecto puede retrasarse), y las relaciones que sostienen unas con otras; 3) el criterio para definir una situación normal es que ésta cumpla con lo planeado en la definición del proyecto, es decir, que no presente *discrepancias* con lo estipulado de manera teórica; 4) la frecuencia está dada mediante estipular aquellas fechas que son determinantes para el desarrollo del proyecto (los entregables, por ejemplo); 5) el manejo de casos especiales se da con respecto a las especificaciones del proyecto (es decir, con base a cada proyecto en particular). Por tanto, la variable restante es aquella que define el *cómo* se realizará el monitoreo; es decir, *de qué manera se hará la detección de discrepancias temporales existentes en las actividades concernientes al proyecto durante una revisión a éste*. El enunciado anterior se puede ver como el problema específico a abordar.

1.2. Objetivo

El objetivo principal de la presente investigación se centra en proveer un mecanismo de monitoreo basado en Razonamiento Temporal que sea capaz de detectar las discrepancias de origen temporal (atrasos, adelantos) que surjan en las actividades a lo largo del desarrollo de un proyecto. Asimismo, se busca *explorar* posibilidades de razonamiento que pudiesen llegar a proveer enfoques más flexibles a lo que convencionalmente se plantea.

Ahora bien, para lograr el objetivo general planteado, se han formulado los siguientes objetivos específicos:

- Encontrar una representación temporal que cubra las necesidades establecidas en el objetivo general (detectar discrepancias de origen temporal en un proyecto). Dicha representación se encuentra en función de buscar alternativas tanto para simbolizar la información como para manejarla, evaluarlas y decidir cuál es la más apropiada.
- Encontrar una herramienta que maneje la representación temporal anteriormente establecida. A la par de identificar el enfoque adecuado de RT que se pueda utilizar con el fin de cumplir el objetivo, es necesario también contar con una herramienta que sirva para su manipulación.
- Desarrollar un algoritmo de monitoreo que pueda trabajar en conjunto con el enfoque y herramienta seleccionados para detectar las discrepancias temporales.
- Aplicar el mecanismo construido a un conjunto de casos de prueba para ver su comportamiento y decidir si es capaz de detectar los atrasos, adelantos y situaciones normales presentados en dichas pruebas.

1.3. Hipótesis

Debido a que se plantea realizar una prueba de concepto en cuanto a la aplicación de Razonamiento Temporal hacia el proceso de monitoreo de las actividades de un proyecto, la hipótesis de esta investigación se define como: *Es posible construir un mecanismo que utilice el Razonamiento Temporal para realizar el monitoreo de un proyecto, con el fin de detectar situaciones que no corresponden a lo planeado, en cuanto al tiempo de ejecución de las actividades se refiere.*

Asimismo, se busca responder a las siguientes preguntas:

- ¿Qué enfoque de Razonamiento Temporal es el más adecuado para llevar a cabo el proceso de monitoreo?
- ¿Qué herramienta se puede utilizar para la manipulación de las primitivas y relaciones temporales incluidas en el enfoque seleccionado?

- ¿Cómo se puede desarrollar un algoritmo de monitoreo que integre el enfoque y herramienta seleccionados para detectar las discrepancias temporales durante el desarrollo de las actividades?
- ¿Puede el mecanismo construido ser capaz de detectar los atrasos, adelantos y situaciones normales presentados en un conjunto específico de pruebas?

1.4. Alcance

El presente trabajo de investigación se ha limitado en varios aspectos. En primera instancia, el trabajo está enfocado a la *dimensión temporal* que comprende a un proyecto; es decir, otros puntos tales como costo, recursos, alcance y calidad quedan fuera del alcance de la investigación. Asimismo, a pesar de que estos puntos son relevantes se ha optado por solamente cubrir el aspecto del *tiempo*. De igual manera, aunque existen varios procesos involucrados con la administración de un proyecto—planeación, organización, ejecución, control, manejo de riesgos, manejo de equipo de trabajo, etc.—únicamente se ha considerado el proceso de monitoreo de un proyecto, el cual incluye la observación de la ejecución de un proyecto para identificar problemas potenciales y equivale a contrastar el desempeño presente con respecto al plan [23].

Por otra parte, en cuanto al Razonamiento Temporal se refiere, conviene destacar que se ha optado por trabajar con lo ya establecido, encontrando para ello aquellos enfoques que mejor cubran los objetivos planteados. De hecho, como se verá más adelante, para fines de la presente investigación básicamente se utilizó el álgebra de intervalos de tiempo y también se hizo uso de la representación que proveen los puntos de tiempo (sin razonar sobre ellos).

Con respecto a la *red de actividades* a utilizar a lo largo de la investigación, ésta es de tipo AON (*Activity On Node*), con dependencias obligatorias y solamente del tipo *Fin a Inicio* (*Finish to Start*) y sin actividades de retroalimentación. Asimismo, el método de análisis sobre la red es el CPM (*Método de la Ruta Crítica*), considerando para ello todo lo que el uso de este método implica (por ejemplo, el hecho de que las duraciones de las actividades son determinísticas).

1.5. Aportación

La principal aportación de este trabajo va dirigida hacia el campo de estudio de aplicaciones de Razonamiento Temporal, en cuanto a que provee una aplicación en la cual éste puede ser utilizado.

Siguiendo esta misma línea, se presenta un mecanismo que implementa, manipula y razona sobre una representación basada en el tiempo con el fin de auxiliar un proceso como lo es el monitoreo de las actividades de un proyecto. En este sentido, se está ofreciendo una *alternativa* de monitoreo al considerar a las actividades de dicho proyecto como intervalos

entrelazados cuyas relaciones—obtenidas mediante realizar inferencias tomando como punto de partida la precedencia existente en la red—deben conservarse para que prevalezca una ejecución que va *a tiempo*.

1.6. Estructura del documento

El presente trabajo se encuentra organizado de la siguiente manera. En el Capítulo 2, se muestran los antecedentes pertinentes a Administración de Proyectos y Razonamiento Temporal. En el Capítulo 3, se describe el mecanismo propuesto tanto en estructura (de manera intuitiva y de manera formal) como en su funcionamiento (se explica el algoritmo que lo implementa y se sigue mediante un ejemplo). En el Capítulo 4, se abordan las pruebas realizadas sobre el mecanismo: cómo se seleccionaron, la descripción de los casos de prueba y los resultados obtenidos. En el Capítulo 5, se presentan las conclusiones que se derivaron de este trabajo y de igual forma se plantean trabajos futuros.

Capítulo 2

Marco Teórico

El presente capítulo tiene como propósito el cubrir algunos aspectos de temas relacionados con la investigación realizada, los cuales se han considerado como relevantes, de tal manera que se propicie un mejor entendimiento del mecanismo de monitoreo propuesto posteriormente. El primero de estos temas es *Administración de proyectos*; se abordan las características de un proyecto, el proceso de seguimiento (monitoreo) y la administración del tiempo, básicamente. El otro tema es *Razonamiento Temporal*; se tratan las principales álgebras que lo componen (con especial énfasis en el *álgebra de intervalos*) y también se hace mención de las herramientas que trabajan con este tipo de razonamiento.

2.1. Administración de proyectos

2.1.1. Características de un proyecto

Un *proyecto* es un esfuerzo *temporal* llevado a cabo con el propósito de crear un producto, servicio y resultado *único*.

Conviene destacar que el atributo temporal se atribuye debido a que todo proyecto cuenta con *un inicio y fin definidos*. El fin se considera alcanzado cuando los objetivos del proyecto se han cumplido o se vuelve claro que éstos no se podrán cumplir.

Asimismo, otro rasgo importante en cuanto a proyectos se refiere es que éstos consisten de una *elaboración progresiva*. Es decir, el desarrollo se da en etapas incrementales y durante la ejecución se van teniendo resultados y especificaciones cada vez más claros y más completos.

Ahora bien, debido a su naturaleza temporal, un proyecto tiene definido un *ciclo de vida*, el cual ocurre en varias etapas [27]. Éstas se muestran en la Figura 2.1.

Las tareas que se ejecutan en cada etapa del ciclo son las siguientes:

- **Diseño conceptual**
  - Metas

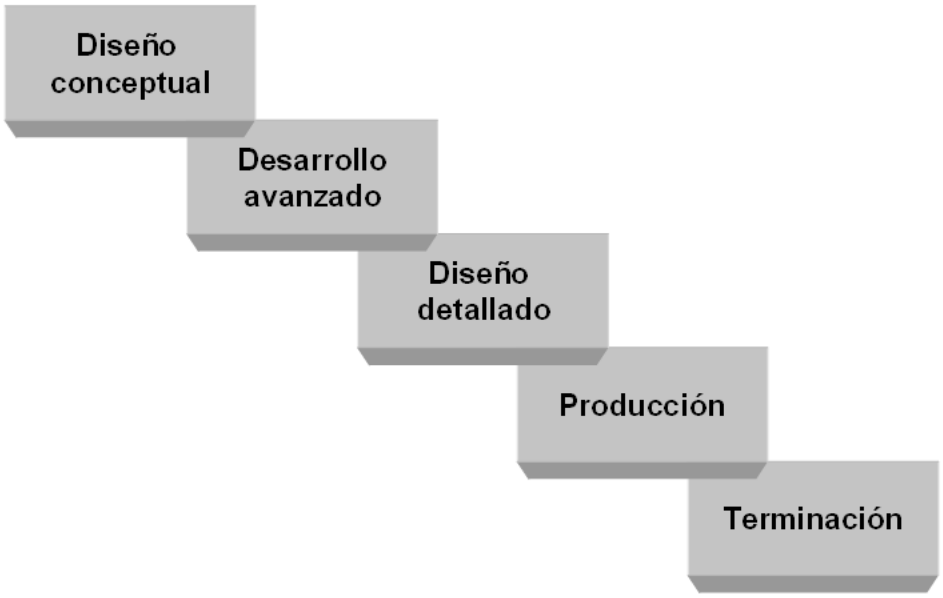

Figura 2.1: Ciclo de vida de un proyecto.

- Alcance
- Baseline
- Requerimientos
- Factibilidad
- Deseabilidad

- **Desarrollo avanzado**
  - Plan
  - Presupuesto
  - Calendarización
  - Propuesta
  - Compromiso de administración

- **Diseño detallado**
  - Definición de responsabilidades
  - Conformación de equipos de trabajo
  - Estructura organizacional
  - Plan detallado
  - Comienzo

- **Producción**

- Administración
- Mediciones
- Control
- Actualización y re-planeación
- Resolución de conflictos

- **Terminación**
  - Cierre
  - Documentación
  - Sugerencias de mejora
  - Reasignación de recursos
  - Disolución de equipos de trabajo

2.1.2. Proceso de administración de proyectos

La *administración de proyectos* se define como la aplicación de conocimiento, habilidades, herramientas y técnicas hacia las actividades del proyecto con el fin de cumplir con los requerimientos impuestos por éste. La administración de proyectos se logra mediante la integración de los procesos de iniciación, planeación, ejecución, monitoreo, control y cierre.

La administración de proyectos *per sé* incluye las siguientes actividades:

- Identificar los requerimientos
- Establecer objetivos claros y alcanzables
- Mantener en balance las demandas competentes entre calidad, alcance, costo y tiempo
- Adaptar las especificaciones, planes y enfoques a las diferentes necesidades y expectativas de los diversos involucrados (*stakeholders*) en el proyecto

El alcance, costo y tiempo del proyecto se puede definir como una "triple restricción" [23]; esto implica que cada una de estas *dimensiones* debe mantenerse en balance (o, en el último de los casos, en un nivel "aceptable" para los involucrados) con respecto a las otras, y al mismo tiempo también conlleva el hecho de que cuando se trabaje en pro de una sola de ellas, esto generalmente será a expensas de las demás. En consecuencia, cualquier acción que repercuta en una también lo hará en las demás, y esto afectará—para bien o para mal—el desempeño (calidad) del proyecto. Es por ello, que una adecuada administración es necesaria. Si los tres factores se encuentran balanceados, es decir, el proyecto produce los *resultados requeridos* de acuerdo a su *alcance*, sale *a tiempo* y no sobrepasa el *presupuesto*, en general se puede decir que el proyecto es exitoso.

Proceso de Monitoreo y Control

En general, los procesos pertinentes a la administración de un proyecto se clasifican en cinco diferentes grupos [23]:

- Grupo de *Apertura*
- Grupo de *Planeación*
- Grupo de *Ejecución*
- Grupo de *Seguimiento (monitoreo) y Control*
- Grupo de *Clausura*

El grupo de seguimiento y control consta de procesos llevados a cabo para observar la ejecución del proyecto, con el fin de *identificar* problemas potenciales y proveer recomendaciones acerca de *acciones preventivas* y/o *correctivas* que se pueden realizar para asegurar que se cumpla con el alcance bajo las restricciones de costo y tiempo. De esta manera, la tarea de monitoreo se centra en medir y analizar el desempeño del proyecto para *encontrar variaciones con respecto al plan establecido* y de estar al pendiente de aquellos factores que pudiesen afectar el desempeño, mientras que la tarea de control se encarga de manejar los cambios efectuados a lo largo del proyecto, recomendar la toma de acciones para corregir inconvenientes y administrar las actualizaciones que surgen a raíz de éstos.

Algunos de los aspectos dentro del proyecto que requieren de los procesos de seguimiento y control son los siguientes:

- Alcance
- Presupuesto (costos)
- Calendario (tiempo)
- Calidad
- Riesgos
- Equipo de trabajo
- Desempeño
- Trabajo

Administración del tiempo en un proyecto

La *administración del tiempo en un proyecto* se refiere a los procesos requeridos para lograr completar a tiempo el proyecto.

Las actividades involucradas en la administración del tiempo en el proyecto incluyen:

Definición de actividades.- Identificación de tareas específicas que deben realizarse para lograr producir los entregables del proyecto.

Secuenciación de actividades.- Identificación y documentación de las dependencias existentes entre las actividades previamente definidas.

Estimación de los recursos de las actividades.- Estimación de tipo y cantidad de recursos requeridos por cada actividad a desarrollar.

Estimación de la duración de las actividades.- Estimación de la cantidad de periodos de trabajo necesarios para completar individualmente las actividades.

Desarrollo de la calendarización.- Análisis de la secuenciación, duración, requerimientos de recursos y restricciones para la producción de un calendario.

Control del calendario.- Control de cambios al calendario establecido.

Para realizar la *Definición de las actividades*, es necesario tener en cuenta varios factores, tales como el alcance del proyecto y la descomposición de tareas—*Work Breakdown Structure*, o WBS por sus siglas en inglés. Un ejemplo de ésta última se muestra en la Figura 2.2 [6]. Como resultado de esta etapa, se realiza la lista de actividades necesarias (las cuales se componen de un tamaño "manejable" [19]) junto con sus atributos y se definen los *milestones* o entregables del proyecto.

En cuanto a la *Secuenciación de actividades*, ésta se puede realizar una vez que se ha estipulado la definición de las mismas. Para completar esta tarea, se requiere establecer las dependencias existentes entre las actividades y producir una representación donde se muestre el flujo de trabajo del proyecto. A este diagrama se le conoce como *Red de actividades*.

Algunas características de la Red de actividades son las siguientes:

- El flujo de la red va de izquierda a derecha (cuando no se consideran actividades de retroalimentación)
- Las flechas indican precedencia y flujo, y pueden cruzarse
- Los nodos representan a las actividades. Éstas se pueden adicionalmente clasificar en:
    - Fusionadoras.- Reciben el flujo de dos ó más actividades

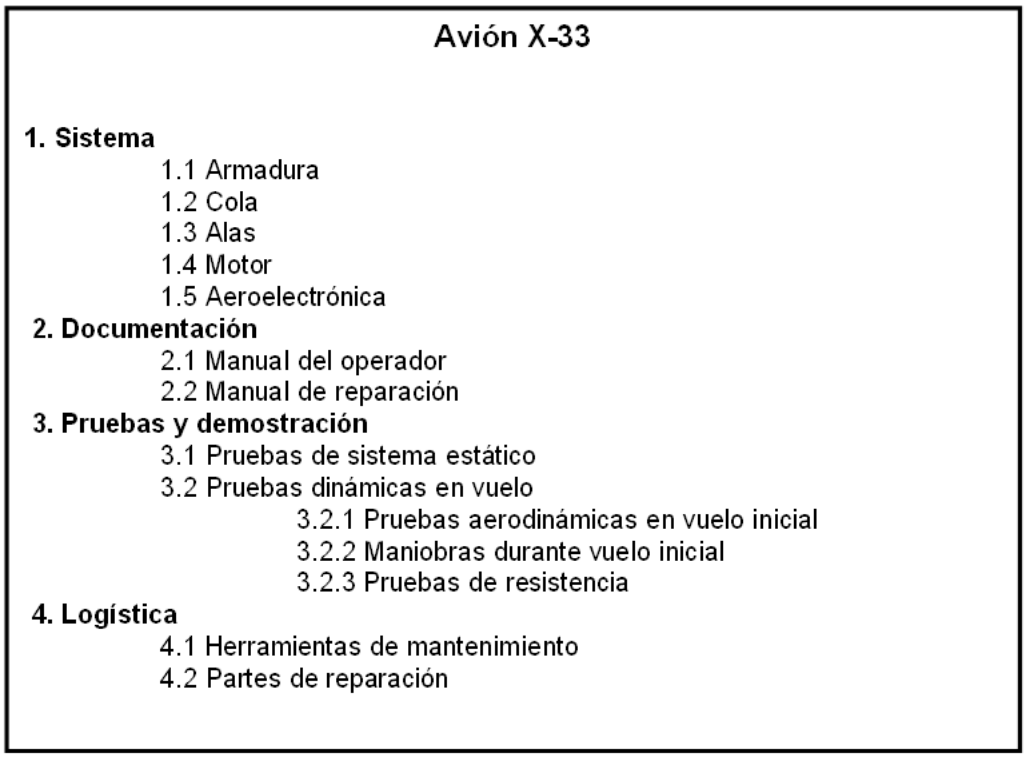

Figura 2.2: Ejemplo de descomposición de tareas (WBS).

  - De ruptura.- Direccionan el flujo hacia dos ó más actividades
  - Paralelas.- Se pueden realizar de manera concurrente

- No se permiten ciclos
- No se permiten estatutos condicionales
- Se sugiere un nodo de inicio cuando existen múltiples actividades de inicio. Igualmente, se recomienda poner un nodo de terminación cuando hay varias actividades que terminen el proyecto.

Existen dos enfoques para representar una red de actividades: *Activity-On-Arc* (AOA, actividad en arco) y *Activity-On-Node* (AON, actividad en nodo). Estas representaciones también reciben los nombres de Método de Diagrama de Precedencias (PDM, por sus siglas en inglés) y Método de Diagrama de Arcos (ADM), respectivamente. La diferencia radica en que el enfoque AOA representa los eventos involucrados en el proyecto como los nodos, y los arcos se utilizan para referirse a la ejecución de las actividades (las cuales se dan entre un evento y otro), mientras que AON visualiza los nodos como las actividades y a los arcos como las relaciones de precedencia que existen entre éstas. El enfoque más utilizado actualmente es el de AON.

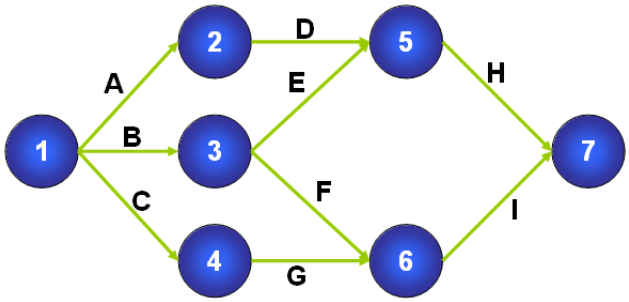

Figura 2.3: Representación Actividad en Arco (AOA).

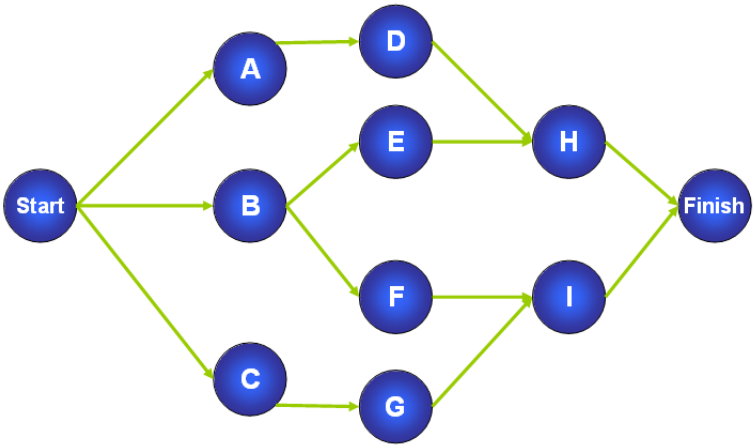

Figura 2.4: Representación Actividad en Nodo (AON).

Asimismo, existen varios tipos de dependencia que se pueden bosquejar en una red de actividades. Éstos son:

1. Final-inicio (*Finish to start*)
2. Final-final (*Finish to finish*)
3. Inicio-inicio (*Start to start*)
4. Inicio-final (*Start to Finish*)

Las dependencias anteriormente mencionadas se muestran de manera gráfica en la Figura 2.5. Adicionalmente, estos tipos se ilustran con más detalle en [16].

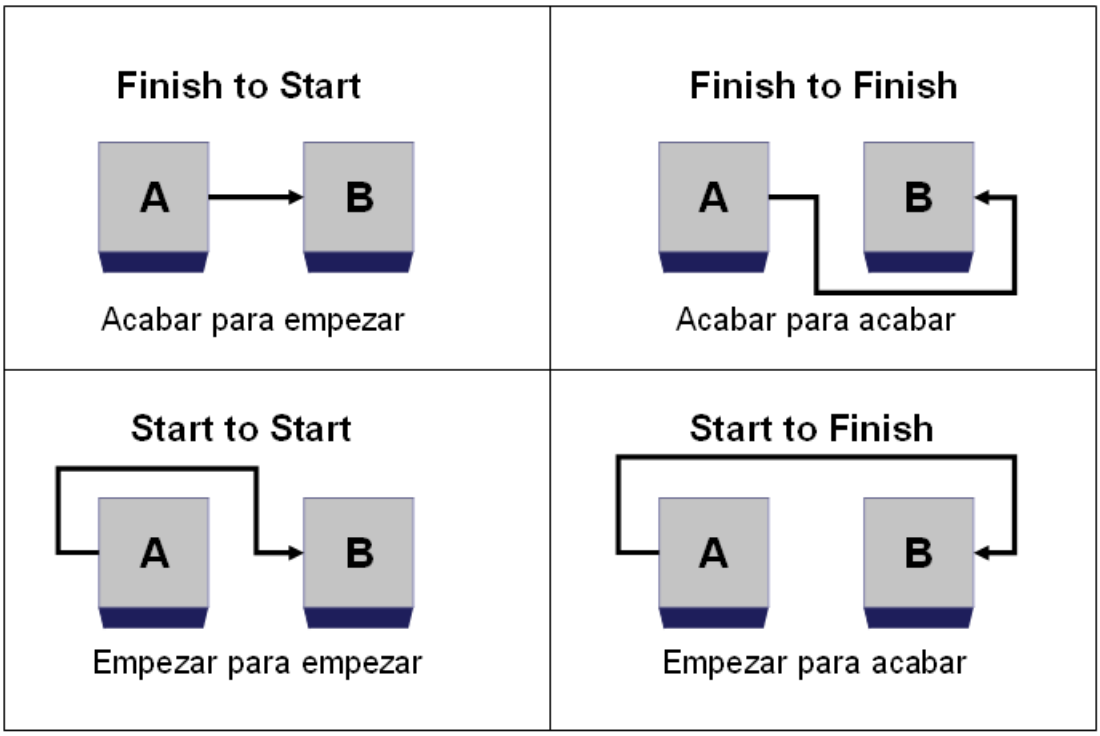

Figura 2.5: Tipos de dependencias.

Cabe destacar que la mayoría de las dependencias descritas en un proyecto son *obligatorias*. Sin embargo, pudiese ser que éstas representen más bien relaciones de precedencia "suaves". [23]

Ahora bien, con referencia a la *Estimación* tanto de recursos como de duraciones, existen varios métodos que se pueden utilizar. Por ejemplo, una técnica es el análisis de datos estadísticos, así como la estimación tipo *bottom-up* (obtener los requerimientos de tiempo y recursos mediante conjuntar las necesidades de cada tarea en particular). De hecho, para la determinación de las duraciones en las actividades, se puede realizar un estimado de "tres puntos", el cual consiste en hacer una suma ponderada del tiempo más probable (tm, tiempo que normalmente se invierte en realizar la actividad), el tiempo pesimista (tp, tiempo que tardaría la actividad en las condiciones más adversas) y el tiempo optimista (to, tiempo mínimo en llevar la actividad a cabo si todo sale excepcionalmente bien) en el que se realiza una actividad. Dicho cálculo generalmente se hace con la fórmula $\frac{(tp+4tm+to)}{6}$ [8].

Tras realizar las estimaciones pertinentes, se elabora el *calendario*. Para su desarrollo y control, actualmente se cuenta también con varias técnicas:

- Diagrama de Gannt
- PERT (*Program Evaluation and Review Technique*)

- Método de la Ruta Crítica (CPM)

El Método de la Ruta Crítica (*Critical Path Method o CPM*, por sus siglas en inglés) se define como una técnica de análisis sobre una red de actividades, en la cual se calculan *fechas tempranas*, así como *fechas tardías* de inicio y terminación de forma *teórica*. Para ello—sin tomar en cuenta las limitaciones impuestas por la cantidad de recursos disponibles—se realiza un análisis hacia *adelante*, seguido por un análisis hacia *atrás* a través de los caminos ("rutas") de la red. De este modo, se determinan los periodos de tiempo en que se deben programar o calendarizar las actividades, dada su duración, relaciones, tipos de dependencias y otras restricciones. Ahora bien, debido a que las duraciones de las actividades son intrínsecamente diferentes, podrán existir lapsos de tiempo en los que el inicio de una actividad puede ser pospuesto sin causar atrasos en el proyecto. Cuando este lapso u "holgura" es inexistente, la actividad se vuelve crítica, pues cualquier retraso causará que el proyecto no salga a tiempo. Es así que aquella ruta en la cual ninguna de las actividades posee holgura se denomina la *Ruta Crítica* [23]

Cabe destacar que, al trabajar con este método, se considera que la duración de las actividades es determinística. Es decir, no existe lugar a probabilidades e incertidumbre. [9] describe con mayor detalle esta clase de redes.

Para tener un concepto más claro de los componentes principales, a continuación se muestran algunas de las definiciones más importantes dentro del método CPM:

Actividad Asignación que debe ser llevada a cabo como parte del proyecto. Cuenta con una duración estimada y tiene fechas establecidas para su inicio y terminación.

Evento Punto de tiempo en el cual una actividad comienza o termina. Por tanto, cada actividad tiene asociados dos eventos: comienzo y terminación.

Milestone (entregable) También conocido como punto de revisión, el milestone es un evento(fecha)contra el cual se mide el avance del proyecto; provee una estructura de control del progreso. Asimismo, también puede representar un punto de toma de decisión, el cual requiere ser pasado para poder continuar con actividades posteriores (por ejemplo, una firma por parte del cliente). Por definición, los milestones tienen una duración de cero.

Fecha temprana de inicio (es) Fecha desde la cual se puede comenzar una actividad.

Fecha temprana de terminación (ef) Fecha en la cual finaliza una actividad si comenzó en su fecha temprana de inicio.

Fecha tardía de inicio (ls) Fecha en la que a más tardar debe comenzar una actividad para evitar retrasos.

Fecha tardía de terminación (lf) Fecha en la que a más tardar debe finalizar una actividad para evitar retrasos.

Holgura (fl) Cantidad de tiempo que existe entre las fechas de inicio o las fechas de terminación.

Ruta crítica Secuencia de actividades que es la de más larga duración. Las actividades que forman parte de la ruta crítica tienen una holgura de cero.

Para calcular los tiempos tempranos y tardíos de las actividades, primero es necesario tener en cuenta las siguientes relaciones entre los tiempos y duraciones:

$$es_i = e_e$$

$$ef_i = es_i + d_i$$

$$lf_i = l_l$$

$$ls = lf_i - d_i$$

$$fl_i = lf_i - ef_i$$

donde:

e_e es el punto de tiempo más temprano en el cual la actividad i puede ser comenzada.
l_l es el punto de tiempo más tardío en el cual la actividad i puede ser finalizada.

Considerando estos datos, el *análisis hacia adelante* en el diagrama de precedencia se hace con las siguientes fórmulas:

$$es_i = argmax_{j \in PA_i}(ES_j + d_j)$$

$$ef_i = ES_i + d_i$$

donde:

PA_i es el conjunto de actividades que preceden a la actividad i directamente.

En consecuencia, en el *análisis hacia atrás*, los tiempos de inicio y terminación se construyen con:

$$lf_i = argmin_{j \in SA_i}(LF_j - d_j)$$

$$ls_i = LF_i - d_i$$

donde:

SA_i es el conjunto de actividades que suceden a la actividad i directamente.

Por ejemplo, para la red en la Figura 2.4, se describe la tabla de cálculos para obtener los tiempos de inicio y terminación [19] en el Cuadro 2.1. Para ilustrar mejor cómo se aplicaron las fórmulas para encontrar dichos datos, considérese la actividad I.

$$\begin{aligned} es_I &= argmax(20, 28) \\ &= 28 \end{aligned}$$

$$\begin{aligned} ef_I &= 28 + 6 \\ &= 34 \end{aligned}$$

$$\begin{aligned} lf_I &= argmin(40) \\ &= 40 \end{aligned}$$

$$\begin{aligned} ls_I &= 40 - 6 \\ &= 34 \end{aligned}$$

$$\begin{aligned} fl_I &= 40 - 34 \\ &= 6 \end{aligned}$$

Actividad	**Duración**	**es**	**ef**	**ls**	**lf**	**fl**
A	10	0	10	0	10	0
B	7	0	7	7	14	7
C	12	0	12	6	18	6
D	18	10	28	10	28	0
E	14	7	21	14	28	7
F	13	7	20	21	34	14
G	16	12	28	18	34	6
H	12	28	40	28	40	0
I	6	27	33	34	40	6

Cuadro 2.1: Datos de actividades pertenecientes a la Figura 2.4

Finalmente, la red que contempla la ruta crítica (obtenida a partir de los datos proporcionados por las actividades) se muestra en la Figura 2.6.

Todos los elementos antes mencionados conforman las bases en cuanto a la administración de un proyecto se refiere.

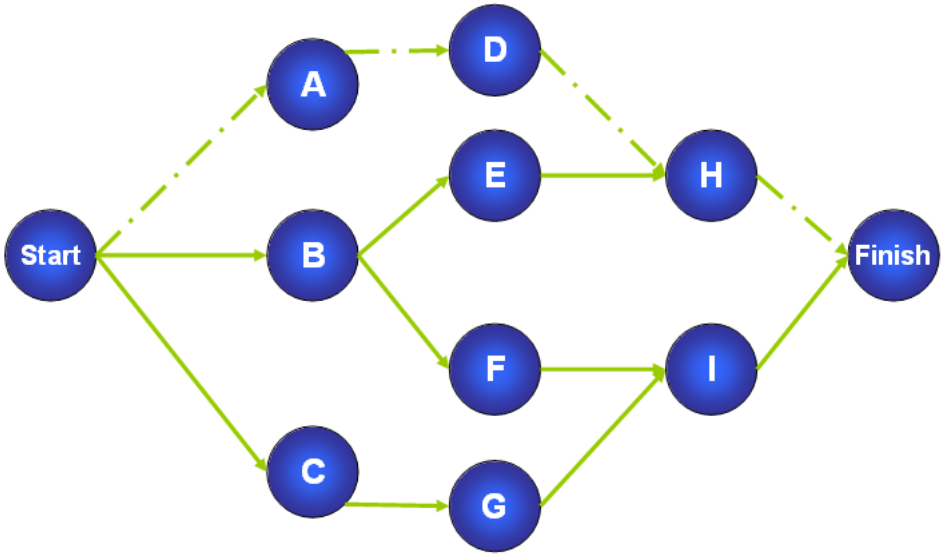

Figura 2.6: Red de actividades con RC

2.2. Razonamiento Temporal

El *Razonamiento Temporal* (abreviado como RT) consiste en formalizar la noción del tiempo y proveer medios para representar y razonar sobre aspectos temporales del conocimiento [28]. A pesar de la importancia del tiempo en el mundo real, este concepto no se comenzó a explorar en la IA sino hasta la década de los 80's del siglo XX (es decir, hace aproximadamente 25 años). Algunos de los investigadores con mayores contribuciones en esta área son James F. Allen, D. McDermott, James van Beek y Henry Kautz.

Algunas aplicaciones generales [28]:

- Planeación
- Supervisión de procesos industriales
- Comprensión del lenguaje natural

Otras aplicaciones se pueden encontrar en [22] y [3].

A pesar de que existen diferentes maneras de representar y razonar sobre el tiempo, existen algunas tendencias con respecto a la utilización de estas representaciones. En general, se hace mayor uso de las llamadas "álgebras temporales". Éstas trabajan sobre *entidades* (puntos, intervalos) que se encuentran *relacionadas* y realizan *operaciones*—las cuales proveen mecanismos de inferencia—sobre estas relaciones.

2.2.1. Álgebra de intervalos

El álgebra de intervalos, introducida por James F. Allen, considera como su primitiva temporal al *intervalo* y denota trece relaciones básicas que pueden darse entre un par de estas entidades.

En primer lugar, un *intervalo*—definido de esta manera por el mismo Allen—, se puede ver como un par ordenado de puntos en el cual el primero es menor al segundo (poner cita de maintaining). Por ejemplo, si A es un intervalo, entonces su punto de inicio (el cual de ahora en delante será conocido como A^-) va *antes* que su punto de terminación (que será consiguientemente llamado A^+), de tal suerte que $A^- < A^+$.

Viéndolo en términos de conjuntos, un intervalo I es equivalente a un conjunto que contiene los elementos $\{pi, P, pt\}$, donde $p_i = I^-$ y $p_t = I^+$. P describe al conjunto de puntos intermedios, donde $\forall p\ p \in P \Rightarrow p > p_i \wedge p < p_t$. De hecho, también es posible ver que los puntos intermedios siguen un orden en el cual $\forall p_x\ p_x \in P \Rightarrow p_{x-1} < p_x < p_{x+1}$. Cabe destacar que la cardinalidad de P está en función de la "precisión"otorgada al intervalo, por lo cual ésta es variable, dependiendo del dominio de aplicación. Por ejemplo, si vemos una hora como intervalo de tiempo, es posible establecer diferentes granularidades, de tal forma que para

algunos dominios sea conveniente ver a esta entidad como una sucesión de 60 minutos y para otros, como una sucesión de 3600 segundos (donde minutos y segundos se consideran como puntos de tiempo).

De esta forma, un intervalo se comprende como una *sucesión* discreta de puntos que se encuentran ordenados.

Según Allen, existen 13 diferentes relaciones temporales (de las cuales 6 son relaciones inversas) entre intervalos de tiempo [20]:

Relación	**Símbolo**	**Inverso**	**Ilustración**
X *antes que* Y	b	$b-$	XXX YYY
X *igual que* Y	e	e	XXX YYY
X *encuentra a* Y	m	$m-$	XXXYYY
X *se empalma con* Y	o	$o-$	XXX YYYY
X *durante* Y	d	$d-$	XXX YYYYY
X *empieza* Y	s	$s-$	XXX YYYYY
X *termina* Y	f	$f-$	XXX YYYYY

Cuadro 2.2: Relaciones Temporales de Allen

De esta forma, contamos con las relaciones: antes (b, también denotado como $<$), igual (e ó $=$), encuentra a (m), se traslapa con (o), durante (d), empieza (s), termina (f), después (a, bi, $>$), es encontrada por ($m-$, mi), es traslapada por ($o-$, oi), es empezada por ($s-$, si), es terminada por ($f-$, fi) y contiene ($d-$, di).

Conviene destacar que estas relaciones son disjuntas entre sí, y cuando se utilizan de manera individual son conocidas como *relaciones simples*. Sin embargo, existen casos en los

que—por la información proporcionada—se tendrán que construir *conjuntos* de relaciones. Por ejemplo, piense en la siguiente sentencia: "Pedro realizará las diapositivas de la presentación mientras que Juan termina el prototipo.". La relación más intuitiva sería (considerando P como el intervalo en el que Pedro realiza las diapositivas y J como el intervalo en el cual Juan termina el prototipo) P o J. Sin embargo, pudiera ser que Juan y Pedro comiencen sus actividades al mismo tiempo y Pedro acabe primero, en cuyo caso la relación temporal sería (P s J), o también pudiese darse una situación en la cual Pedro comenzó antes que Juan, pero ambos terminaron al mismo tiempo, por lo que la relación correspondería a (P f J). Más aún, puede ser que Pedro comience primero y termine después que Juan, lo cual equivale a decir que (P d J), o inclusive puede darse el caso en que ambos ocupen exactamente el mismo tiempo para hacer las actividades (P e J). Por lo tanto, mientras que no se tenga la información suficiente para determinar cuál de las relaciones toma lugar, lo correcto es representar la situación mediante un *vector de relaciones*. De esta forma, el enunciado se puede/debe expresar como (P (o, e, s, f, d) J).

Asimismo, mediante la utilización de vectores de relaciones, es posible introducir operaciones que faciliten el proceso de *inferencia*, con el cual se obtienen relaciones entre intervalos que no se encuentran directamente relacionados, lo cual se explica a continuación.

Inferencia con intervalos de tiempo

Al conocer los componentes del álgebra de intervalos (los intervalos y las relaciones que sostienen individualmente entre ellos), es posible describir el proceso de inferencia, el cual sirve—como cualquier proceso de este mismo tipo—principalmente el propósito de derivar nuevos datos a partir de los hechos dados. Cabe destacar que la inferencia pertinente a las relaciones temporales conformadas por intervalos hace uso de dos operaciones básicas entre vectores de relaciones: *suma* y *multiplicación*.

La *suma* entre dos vectores $V1$ y $V2$ es simplemente la operación de intersección entre conjuntos (los vectores deben referirse al mismo par de intervalos, o de otra forma la suma no es posible). Por ejemplo, en 2.3, tenemos cómo se suman los vectores de las relaciones (A (b, o, m) B) y (A (o, s, d) B).

$$V1 = (b, o, m)$$
$$V2 = (o, s, d)$$
$$V1 + V2 = (o)$$

Cuadro 2.3: Ejemplo de suma de vectores de relaciones

Por otra parte, la *multiplicación* (llamada también *composición de vectores*) se da entre un vector $V1$ que existe entre un par de intervalos A y B y un vector $V2$ que se da entre B y

C. Lo que se trata de encontrar es el vector resultante que relaciona a A con C, para lo cual se calcula la cerradura transitiva entre las relaciones temporales dadas. Para facilitar este último cálculo, se cuenta con una tabla de transitividad (definida por Allen en referencia), la cual se ilustra en el Cuadro 2.4.

Debido a que la operación de multiplicación es un poco más compleja que la de suma, ésta se explica en una mayor cantidad de pasos (ver Cuadro 2.5).

En el Cuadro 2.5 lo primero que se hizo fue obtener el producto cartesiano (multiplicación entre pares) de $V1$ y $V2$ para encontrar un vector cuyos elementos relacionen los intervalos A y C. Enseguida, por cada par ordenado se busca su entrada en la tabla de transitividad y ésta se agrega como un elemento resultante de la composición de vectores. Finalmente, los resultados se combinan (se eliminan los elementos repetidos, tal como en la unión de conjuntos) para producir el vector final que se obtiene al realizar la multiplicación. Como es posible de apreciar, este vector conforma una *inferencia* realizada sobre un par de intervalos; por tanto, se puede decir que la multiplicación es el componente *esencial* dentro de este proceso.

Para mantener y manejar la información proporcionada acerca de los intervalos y las relaciones temporales que se dan entre ellos, se hace uso de una red. En ésta, los nodos representan los intervalos y los arcos, las relaciones que existen entre ellos. Además de proporcionar una vista gráfica, esta notación ayuda a asegurar que no se presentarán ambigüedades.

Con el afán de mostrar cómo se realiza la inferencia, las operaciones revisadas hasta el momento y la utilización de las redes temporales, a continuación se introduce un ejemplo. Este ejemplo se ilustra en [1].

Se tiene el siguiente enunciado: "John no estaba en la habitación cuando toqué el interruptor para encender la luz". Los intervalos que intervienen en dicha sentencia son los siguientes:

- Tiempo en que John no estaba en la habitación (denotado por R)
- Tiempo en que la luz estaba encendida (L)
- Tiempo en que se tocó el interruptor (S)

De acuerdo a las relaciones posibles entre cada par de intervalos, se obtiene una red temporal como 2.7.

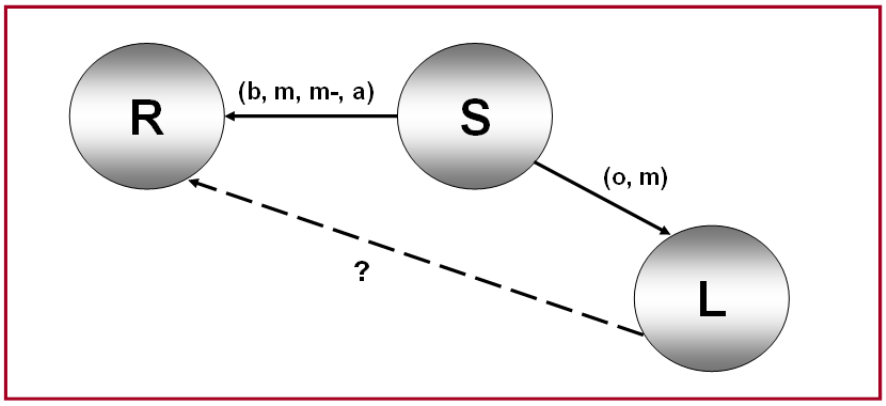

Figura 2.7: Ejemplo interruptor de luz

Ahora bien, si se desea inferir la relación que existe entre L y R (relación que se da entre el tiempo que John estuvo en la habitación y el tiempo en que la luz se encontraba prendida), lo primero que hay que hacer es trazar un camino que lleve desde L hasta R, el cual se obtiene mediante el intervalo S. Las relaciones de tal camino se describen por $(L(o-, m-)S)$ y $(S(a, m, m-, b)R)$ (observe que el vector de relaciones entre L y S se invirtió para que fuera en la dirección contraria). Una vez contando con estos datos, puesto que lo que nos interesa es obtener el vector pertinente a la relación entre L y R, realizamos la multiplicación entre los vectores V_{LS} y V_{SR}. Para ello, se obtiene el producto cartesiano y se aplica la tabla de transitividad.

$$
\begin{array}{ll}
T(o-, b) = & (b, o, m, d-, f-) \\
T(o-, m) = & (o, d-, f-) \\
T(o-, m-) = & (a) \\
T(o-, a) = & (a) \\
T(m-, b) = & (b, o, m, d-, f-) \\
T(m-, m) = & (s, s-, e) \\
T(m-, m-) = & (a) \\
T(m-, a) = & (a)
\end{array}
$$

Finalmente, resta realizar la combinación de resultados, y para ello se efectúa la unión entre los distintos vectores obtenidos, lo cual da el vector resultante $V_{LR} = (a, b, o, m, d-, s, s-, f-, e)$.

Hasta ahí, se ha hecho una inferencia sobre la base de hechos proporcionada. No obstante, ¿qué pasa si se agregan nuevas *restricciones* a la red temporal? Como se puede suponer, el vector de relaciones recién obtenido cambiará para reflejar este conocimiento recientemente introducido. Por ejemplo, si más tarde se agrega el hecho de que $(L\ (o, s, d)\ R)$, entonces el vector V_{RL} actualizado ahora contendrá el resultado de *sumar* el anterior vector con el nuevo introducido. Esto arroja $V_{LR} = (o, s)$. Asimismo, es necesario propagar las restricciones de los arcos restantes en la red (para lo cual hay que realizar de nuevo el mismo procedimiento

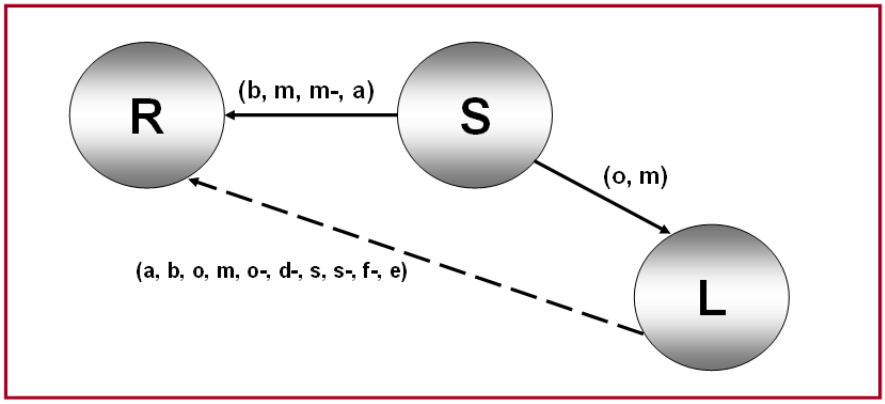

Figura 2.8: Ejemplo interruptor de luz

que se siguió para encontrar V_{LR}). De esta manera, vemos que la el vector de relación entre S y R queda como (b, m). Esto se ilustra en la Figura 2.9.

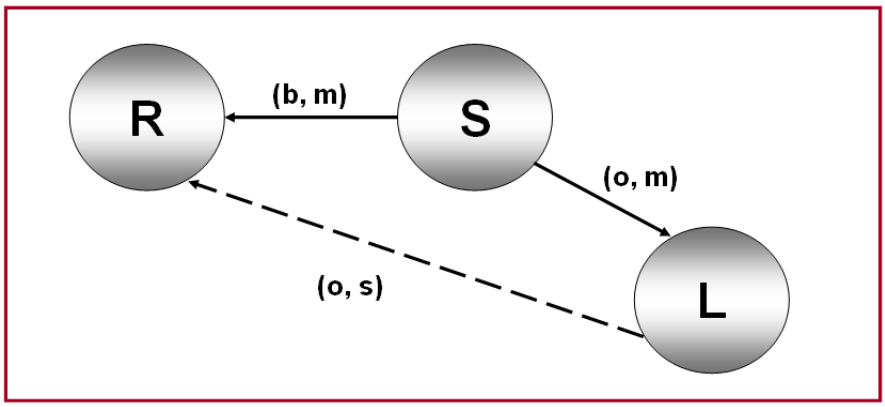

Figura 2.9: Red actualizada después de agregar el hecho $(L(o, s, d)R)$

Debido a que, por cada arco que se agrega a la base de hechos, se constriñen los vectores de relaciones entre los intervalos, el proceso de inferencia se puede modelar como un *problema de satisfacción de restricciones* [20]. Esto último quizá resulta más evidente al realizar la tarea de *propagar* las consecuencias de agregar nueva información a la red de intervalos.

Si bien el algoritmo de propagación de restricciones propuesto por Allen es sólido, se puede decir que éste no es completo en un tiempo polinomial, dado que el probar la consistencia de las restricciones resulta ser NP-difícil [7]. Sin embargo, existen varias maneras de trabajar alrededor de este inconveniente: 1) limitar la cantidad de sentencias utilizadas, 2) aceptar trabajar con un algoritmo polinomial que es incompleto (este algoritmo tiene un orden de $O(n^5)$) o 3) hacer uso de álgebras más "ligeras".

2.2.2. Álgebra de puntos

Mientras que para Allen la primitiva temporal básica es el intervalo, para Kautz y van Beek, ésta resulta ser el *punto* [29]. A pesar de que en un principio pareciera que esta unidad de tiempo es menos intuitiva que los intervalos, ciertamente hay eventos que se expresan de manera más clara utilizando puntos (por ejemplo, los sucesos instantáneos, en los cuales es difícil delimitar los puntos de inicio y fin). Por otra parte, ya que existen menos relaciones posibles entre cada par de puntos, los algoritmos para realizar inferencia resultan más eficientes en cuanto a tiempo de ejecución.

Algunas de las situaciones en las cuales los puntos de tiempo se prefieren sobre otras entidades incluyen las siguientes [28]:

Estados instantáneos.- Al describir los estados de un objeto o situaciones, es común considerar la información como si fuese un punto de tiempo. Ejemplos de esto son proposiciones tales como "La humedad relativa a las 10am es de 90 %", "El valor del dólar con respecto al yen al cierre del día fue de $12.00".

Eventos de *cumplimiento*.- Proposiciones como "La puerta se cerró", "El arma se disparó" y "Llegó la fecha de entrega" se ven intuitivamente como entidades puntuales.

Transiciones.- El pasar de un estado a otro también se considera como un suceso instantáneo, y es más fácil de visualizar como un punto. Por ejemplo, considérese la transición que existe en un pestañeo, o el cambio que se produce en la mano de un robot que pasa de estar cerrada a estar abierta.

El álgebra de puntos considera tres relaciones básicas: antes de ($<$), igual que($=$) y después de ($>$), y siete vectores de relaciones (compárese este número con la cantidad de vectores de relaciones resultantes en el álgebra de intervalos, el cual corresponde a $2^1 3$ y se podrá ver por qué es que el álgebra de puntos es en general más tratable), que corresponden a ($<$), ($=$), ($>$), ($<, =$), ($=, >$), ($<, =, >$), ($<, >$).

La representación puntual, al igual que el álgebra de intervalos, utiliza los operadores de suma y multiplicación para realizar las derivaciones que se dan entre pares de puntos sin aparente conexión. De hecho, el proceso de inferencia es básicamente el mismo. Se ha establecido que la complejidad de este tipo de álgebra (dependiendo de las restricciones que se le impongan) se encuentra entre $O(n^3)$ y $O(n^4)$.

2.2.3. Otras maneras de representar el tiempo

Si bien la utilización de entidades y relaciones temporales facilita el RT, el tiempo como tal puede ser representado de otros modos. Algunos de éstos son fechas absolutas, pseudo-fechas y representaciones basadas en duración [2] [28].

El enfoque de fechas absolutas consiste en adoptar un esquema basado en fechas y representar cada evento como un *timestamp*; la ventaja de este tipo de esquema es que sólo se necesitan hacer operaciones algebraicas simples para obtener datos como la duración de los eventos. Por tanto, los algoritmos generados en este caso son de tiempo constante y la cantidad de espacio utilizado es lineal. Sin embargo, es necesario contar con las fechas absolutas de todos los eventos.

Una representación alternativa es la utilización de pseudo-fechas (numeración de eventos), donde el único dato necesario es el orden en que se suscitan los eventos. Cada vez que se conoce un nuevo evento, se agrega a la base de datos y se acomoda de acuerdo al orden que debe ocupar en el tiempo [1].

Por otra parte, los enfoques basados en duración plantean representar los eventos en un orden parcial y sus relaciones como rangos de duración entre un evento y otro. Con esto, se pretende resolver problemas que involucran el conocimiento de duraciones de eventos.

2.2.4. Semi-intervalos

Hasta cierto punto, es común enfrentarse a situaciones en las que no todo el conocimiento requerido está disponible. Por tanto, es posible que se desconozcan datos como el inicio o fin de uno o más intervalos de tiempo. Esto, obviamente, coloca un obstáculo para aplicar el Razonamiento Temporal, ya que no resulta tan trivial derivar las posibles relaciones temporales entre dos intervalos sin saber exactamente cuándo es que éstos empiezan y/o terminan. No obstante, existe un enfoque que aborda este problema y provee una forma de trabajar con información incompleta o incierta: el uso de semi-intervalos.

Un *semi-intervalo* puede definirse como una entidad temporal que corresponde al inicio o fin de un evento determinado [11]. Ahora bien, estas entidades pertenecen a lo que se denomina *conocimiento burdo*. Este tipo de conocimiento provee fragmentos de información más general (contrario al *conocimiento fino*, el cual dota de datos atómicos y precisos). Sin embargo, su poder de expresión puede llegar a ser igual al de conocimiento fino, cuando se tienen varias piezas para complementar el conocimiento obtenido. Esto se ilustra más adelante ...

Para entender más a fondo el concepto de semi-intervalo, resulta necesario tener en cuenta que las relaciones entre intervalos se dan con referencia a sus puntos iniciales y finales. Por ejemplo, la relación *before* entre dos intervalos A y B se da cuando el punto final de A es mayor al punto inicial de B ($A^+ < B^-$); de esta misma manera, se pueden conformar relaciones más complejas (como *overlaps*) al realizar conjunciones sobre las posibles relaciones entre estos cuatro puntos (A^-, A^+, B^-, B^+). Sin perder de vista lo anteriormente mencionado, es importante familiarizarse con el concepto de *vecindad*.

Ahora bien, con el afán de comprender lo que es una vecindad, convendría entender primero la definición de un *vecino*. Para ello, se dice que dos relaciones entre pares de eventos son vecinos conceptuales si se pueden transformar directamente una en la otra mediante deformar (por deformar nos referimos a tres acciones: acortar, alargar o mover) continuamente los eventos en un sentido topológico [11]. Éntiendase por eventos, los intervalos.

Por ejemplo, las relaciones b (before) y m (meets) son vecinos, ya que si se alarga o mueve uno de los eventos que conforman una de estas relaciones es posible llegar a la otra. Note que esto no sucede con b y o (overlaps), ya que a pesar de que sí resulta factible llegar de una relación a la otra, esto solamente se puede lograr indirectamente (es decir, pasando antes por la relación m). En este sentido, las vecindades entre relaciones temporales se pueden ver como distancias de Hamming de una unidad, donde esta distancia se mide con base a la cantidad de transformaciones realizadas para pasar de una relación a otra.

De la anterior manera, se puede considerar que un conjunto de relaciones entre pares de eventos constituye una vecindad si sus elementos se encuentran conectados por relaciones de vecinos. Por otra parte, las vecindades pueden verse a su vez como nuevas relaciones—pero ahora entre semi-intervalos—las cuales pertenecen al conocimiento burdo. En consecuencia, ya no es necesario pensar en términos de *intervalos* y *relaciones temporales "finas"*, sino que es posible visualizar los eventos y sus nexos en términos de *semi-intervalos* (los cuales, como ya se vio, son intervalos que no se encuentran totalmente definidos) y *relaciones temporales "burdas"* (que simplemente representan un *conjunto* de relaciones posibles entre los intervalos dada la información que se sabe). Las relaciones temporales definidas mediante el uso de semi-intervalos se ilustran en el Cuadro 2.6.

Como se puede ver, gracias a las vecindades, los semi-intervalos y las nuevas relaciones obtenidas es posible razonar en términos generales sobre cuestiones temporales en las cuales no todos los datos están disponibles. De hecho, este proceso es similar al utilizado con álgebra de intervalos, pero con la diferencia de que no se trata con información totalmente conocida. Sin embargo, existe una cuestión primordial que es necesario abordar al tratar con el conocimiento burdo, puesto que toda facilidad tiene un costo. Esto lleva a la pregunta: ¿es posible hacer un razonamiento fino a partir de los semi-intervalos? La respuesta es afirmativa, si se considera que las diferentes piezas de información obtenidas por los semi-intervalos pueden juntarse y así generar datos más particulares.

Como ejemplo de lo anterior, se tienen las siguientes dos relaciones (que pudieron ser obtenidas en distintos tiempos, cuando no toda la información estaba al alcance):

A *ol* B

A *pr* B

Como la relación *ol* comprende las relaciones temporales $(b, m, o, f-, d-)$ (el conjunto de relaciones simples abarcado por cada relación de semi-intervalos se ilustra en el Cuadro 2.7), y a su vez la relación *pr* consiste de las relaciones (b, m), se puede deducir (mediante la intersección de ambos conjuntos de relaciones) que el vector relaciones entre A y B es (b, m).

Esto es en el caso más simple, pero también se pueden obtener relaciones pertenecientes a más de un par de intervalos (mediante utilizar el operador de composición).

R1 / R2	***b***	***a***	***d***	***d–***	***o***	***o–***	***m***	***m–***	***s***	***s–***	***f***	***f–***
a	b	N/A	b o m d s	b	b	b o m d s	b	b o m d s	b	b	b o m d s	b
d	N/A	a	a o- m- d f	a	a o- m- d f	a	a o- m- d f	a	a o- m- d f	a	a	a
d–	b	a	d	N/A	b o m d s	a o- m- d f	b	a	d	a o- m- d f	d	b o m d s
o	b o m d- f-	a o- d- m- s-	o o- d s f d- s- f- e	d-	o d- f-	o- d- s-	o d- f-	o- d- s-	o d- f-	d-	o- d- s-	d-
o–	b	a o- d- m- s-	o d s	b o m d- f-	b o m	o o- d f s d- f- s-	b	o- d- s-	o f-	o d- s	o d	b o m
m	b	a o- m- d- s-	o d s	b	b	o d s	b	f f- e	m	m	o d s	b
m–	b o m d- f-	a	o- d f	a	o- d f	a	s s- e	a	d f o-	a	m-	m-
s	b	a	d	b o m d- f-	b o m	o- d f	b	m-	s	s s- e	d	b o m
s–	b o m d- f-	a	o- d f	d-	o d- f-	o-	o d- f-	m-	s s- e	s-	o-	d-
f	b	a	d	a o- m- d- s-	o d s	a o- m-	m	a	d	a o- m-	f	f f- e
f–	b	a o- m- d- s-	o s d	d-	o	o- d- s-	m	s- o- d-	o	d-	f f- e	f-

Cuadro 2.4: Tabla de transitividad de intervalos

$V1 = (b, m, o)$
$V2 = (b, m)$
$V1 \times V2 = ((b,b),(b,m),(m,b),(m,m),(o,b),(o,m))$ (Producto Cartesiano)
$V1 * V2 = (b, b, b, b, b, b)$ (Resultados tabla transitividad)
$V1 * V2 = (b)$ (Combinación de resultados)

Cuadro 2.5: Ejemplo de multiplicación de vectores de relaciones

Relación	**Símbolo**	**Inverso**	**Ilustración**
X es mayor que Y	ol		XXX????
Y es menor que X		yo	YY
X está inicio a inicio con Y	hh		XXX??
		hh	YYYY
X sobrevive a Y	sv		???????XX
Y es sobrevivido por X		sb	YY
X está final a final con Y	tt		??XXX
		tt	YYYY
X precede a Y	pr		XXX?
Y sucede a X		sd	YYY
X es contemporáneo de Y	ct		?XXX???
		ct	???YYY?
X nace antes de la muerte de Y	bd		XXX?????
Y murió después del nacimiento de X		db	??????YYY

Cuadro 2.6: Relaciones Temporales entre Semi-Intervalos

Relación	**Conjunto que representa**	**Relación con puntos**
ol	e m o f- d-	$A^- < B^-$
hh	s- e s	$A^- = B^-$
yo	d f o- m- a	$A^- > B^-$
sb	b m o s d	$A^+ < B^+$
tt	f- = f	$A^+ = B^+$
sv	d- s- o- m-	$A^+ > B^+$
pr	b m	$A^+ \leq B^-$
bd	b m o f- d- s- e s d f o-	$A^- < B^+$
ct	o f- d- s- e s d f o-	$A^- < B^+, A^+ > B^-$
db	o f- d- s- e s d f o- m-	$A^+ > B^-$
sd	m- a	$A^- \geq B^+$
ob	b m o	$A^- < B^-, A^+ < B^+$
oc	o f- d-	$A^- < B^-, A^+ > B^-$
sc	d- s- o-	$A^- < B^+, A^+ > B^+$
bc	o s d	$A^+ > B^-, A^+ < B^+$
yc	d f o-	$A^- > B^-, A^- < B^+$
ys	o- m- a	$A^- > B^-, A^+ > B^+$

Cuadro 2.7: Conjunto de relaciones representadas por semi-intervalos y representación mediante puntos extremos

2.2.5. Herramientas para trabajar con RT

Para aterrizar la teoría referente a Razonamiento Temporal y poder utilzarla en contextos reales, es necesario contar con herramientas que manipulen las entidades y relaciones temporales, y que sean capaces de actuar como máquinas de inferencia. Dentro de la literatura, se puede encontrar referencia a algunas de éstas:

- **MATS (Metric Allen Time System).-** Herramienta que maneja las relaciones descritas por Allen para resolver problemas temporales basados en restricciones.
- **TimeLogic.-** Máquina de inferencia tipo forward chaining, basada en intervalos de tiempo, la cual trabaja con restricciones [17].
- **Rhet.-** Herramienta basada en TimeLogic, que incluye relaciones absolutas.
- **TimeGraph I y TimeGraph II.-** Sistema de razonamiento temporal, el cual maneja relaciones puntuales, relaciones entre intervalos y fechas absolutas basadas en calendarios y duraciones [12].

La herramienta TimeGraph II (la cual resulta de interés debido a su contribución para el presente trabajo) consiste básicamente de paquetes implementados en el lenguaje LISP (Common LISP), los cuales pueden ser utilizados directamente mediante operación en consola o pueden ser invocados desde cualquier aplicación que corra bajo la plataforma de Allegro Common LISP. Asimismo, TG-II contiene dos interfaces: una para trabajar con puntos de tiempo—establecida de forma predeterminada—y otra para hacerlo con intervalos (cabe destacar que ambos modos no se pueden mezclar). De esta manera, el usuario puede realizar inserciones a la base de conocimiento mediante asertar relaciones entre intervalos o puntos para posteriormente realizar inferencias y consultas sobre esta base.

Internamente, Timegraph II trabaja con una estructura denominada "grafo temporal" (*timegraph*), la cual no es sino la red de entidades temporales (puntos o intervalos, dependiendo del caso) vista anteriormente, cuyos vértices representan las entidades y cuyos arcos denotan las relaciones que hay entre ellos. Este grafo se particiona en conjuntos de *cadenas* (es decir, conjuntos de puntos ordenados de manera lineal), las cuales se recorren al realizar búsquedas para dar respuesta a las consultas presentadas.

Ahora bien, la complejidad bajo la que corre Timegraph II depende de la interfaz que se esté usando. Si ésta es la de puntos, generalmente opera $O(e) \approx O(n^3)$, donde e representa el número de arcos del grafo temporal. Para el álgebra de intervalos, mejora la eficiencia al utilizar técnicas como "reglas de poda", *backtracking* selectivo y "propagación hacia adelante". Resultados experimentales ejecutados sobre el algoritmo usado por TG-II muestran que de hecho éste se comporta de manera polinomial en la práctica. Por otro lado, la complejidad espacial se reporta como lineal, $O(n)$.

Por tanto, al manejar varias representaciones temporales, trabajar en un tiempo razonable, encontrarse disponible y además contar con documentación para su uso, TG-II se convierte en una herramienta a considerar para propósitos de la investigación.

Resumiendo esta sección, se han visto algunos de los componentes más importantes del RT: álgebras (que cubren entidades y operaciones), manejo de información incompleta y herramientas.

Capítulo 3

Mecanismo de monitoreo propuesto

Debido a que la tarea de seguimiento principalmente consiste en *comparar* el desarrollo planeado del proyecto con el desarrollo actual, el pensar en un mecanismo que ejecute esta labor implica plantear un mecanismo de *monitoreo*. Por otra parte, considerando que en una red de actividades éstas a final de cuentas constituyen intervalos de tiempo que se encuentran restringidos por relaciones de precedencia, es posible utilizar el Razonamiento Temporal para inferir otras relaciones que se presentan entre estos intervalos. Por tanto, teniendo en cuenta todo lo previamente establecido: 1) si las relaciones (entre actividades) inferidas mediante la máquina de inferencias del RT se usan como un esquema que indica los nexos que "deberían" de darse de acuerdo a lo planeado, y 2) asimismo con cierta frecuencia se verifica que las actividades cumplan con alguna de estas relaciones, entonces 3) es posible proponer un *mecanismo de monitoreo basado en Razonamiento Temporal* que busque identificar discrepancias entre lo planeado y lo actual en cuanto a la dimensión temporal se refiere.

El presente capítulo tiene dos secciones centrales. La primera de ellas consiste en explicar la conceptualización realizada para definir la estructura del mecanismo; es decir, consiste en establecer los componentes que sirven de base para el entendimiento de éste. Del mismo modo, la segunda trata acerca de exponer el funcionamiento del mismo y mostrar mediante un ejemplo cómo es que trabaja.

3.1. Estructura del mecanismo

3.1.1. Proyecto

Para tener una referencia completa del esquema propuesto, es necesario establecer primeramente una descripción general de cómo se visualiza el concepto de proyecto e introducir los elementos básicos que se están tomando en cuenta para su revisión. Todo lo anterior pretende ayudar a que se comprendan mejor las premisas e implicaciones que se consideran en el mecanismo de monitoreo y su respectiva implementación.

Cualquier proyecto contiene en sí varios componentes. En primera instancia, tiene una *fecha de inicio* y una *fecha de fin* establecidas (de hecho, esta característica es precisamente la que lo distingue de los procesos o trabajos rutinarios). De igual manera, al ser acotado por fechas que determinan su tiempo de ejecución, el proyecto intrínsecamente cuenta con una

duración, la cual se encuentra—por decirlo de alguna manera—distribuida entre las diferentes *actividades* que lo conforman. Consecuentemente, debido a estas propiedades, el proyecto puede verse como un gran intervalo, el cual *contiene*—temporalmente hablando—a todas las actividades estipuladas para completarlo. Esto es algo que posteriormente se irá viendo.

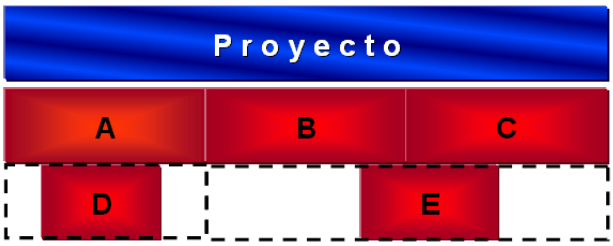

Figura 3.1: Proyecto y actividades vistos como intervalos de tiempo

Ahora bien, para ser manipuladas con mayor facilidad, todas las fechas o *puntos de tiempo* que abarca el proyecto, desde su comienzo hasta su terminación, pueden ser aterrizadas a *tiempos absolutos* (lo cual no constituye algo fuera de lo común en administración de proyectos), siendo así la fecha de inicio igual a cero, la fecha de terminación igual a la duración del proyecto y cualquier fecha intermedia, un número natural que se encuentra entre éstos dos. Este mapeo puede ser realizado mediante una función de conversión (que en este caso denominamos ϕ), la cual toma una fecha y regresa el punto que representa en el proyecto. Como resultado, todos los tiempos del proyecto se manejan bajo este concepto.

Por otra parte, si el proyecto se considera como un intervalo compuesto por n puntos de tiempo, entonces podemos decir que las fechas o puntos de monitoreo son una colección derivada de este conjunto, puesto que la revisión generalmente se hará sólo en ciertas ocasiones que se consideren importantes (aún así, es conveniente hacer notar que pueden existir casos en los que los puntos de monitoreo se hagan en cada fecha correspondiente al mismo, es decir, siempre).

Análogamente, los puntos de revisión (*milestones*)—que representan los entregables del proyecto—, son fechas especiales a considerar y pueden ser visualizados asimismo como otro subconjunto de puntos de tiempo dentro del proyecto.

Otra parte muy importante en cualquier proyecto son las actividades contenidas en el mismo, las cuales son entidades que cuentan con propiedades como duración, tiempo de inicio, etc. De hecho, visto desde la perspectiva del razonamiento temporal, éstas se pueden considerar como *subintervalos* del intervalo que describe el proyecto. Sin embargo, no se puede hablar de éstos como si se encontraran aislados, ya que existen relaciones de *dependencia* entre las actividades, lo cual lleva a que los intervalos sucedan en cierto orden (y esto de alguna manera nos hace ver que existen *relaciones temporales* entre las actividades). En conjunto,

estos dos elementos (actividades y relaciones) es posible visualizarlos bajo una estructura de red, que en términos simples, representa al proyecto en función del tiempo.

Por lo tanto, de manera intuitiva y visto solamente desde la dimensión temporal, un proyecto puede ser conformado por: 1) un punto de inicio y un punto de terminación, los cuales enmarcan en general a un conjunto de puntos de tiempo estipulados para la ejecución, 2) una duración, 3) un conjunto de puntos de revisión (*milestones*), 4) un conjunto de puntos de monitoreo, y 5) un conjunto de actividades relacionadas (representadas mediante una estructura de red) que deben completarse de acuerdo a las precedencias establecidas. Siendo esto último un elemento fundamental y que a su vez se compone de varias otras consideraciones, se procede a explicarlo detalladamente a continuación, enfocándolo hacia la perspectiva del RT.

Debido a que la *Red de actividades* mencionada anteriormente consiste en un digrafo acíclico, donde los nodos representan a las actividades y los arcos a las relaciones de precedencia entre éstas, es posible "aumentar" esta estructura, de tal forma que incluya la información proporcionada por un diagrama temporal [1]. Así, cada actividad—debido a que se encuentra acotada por tiempos de inicio y de terminación—podría considerarse como un intervalo, y cada relación de precedencia (indicada en la red) que se sostiene con otros nodos estaría del mismo modo marcada con etiquetas temporales.

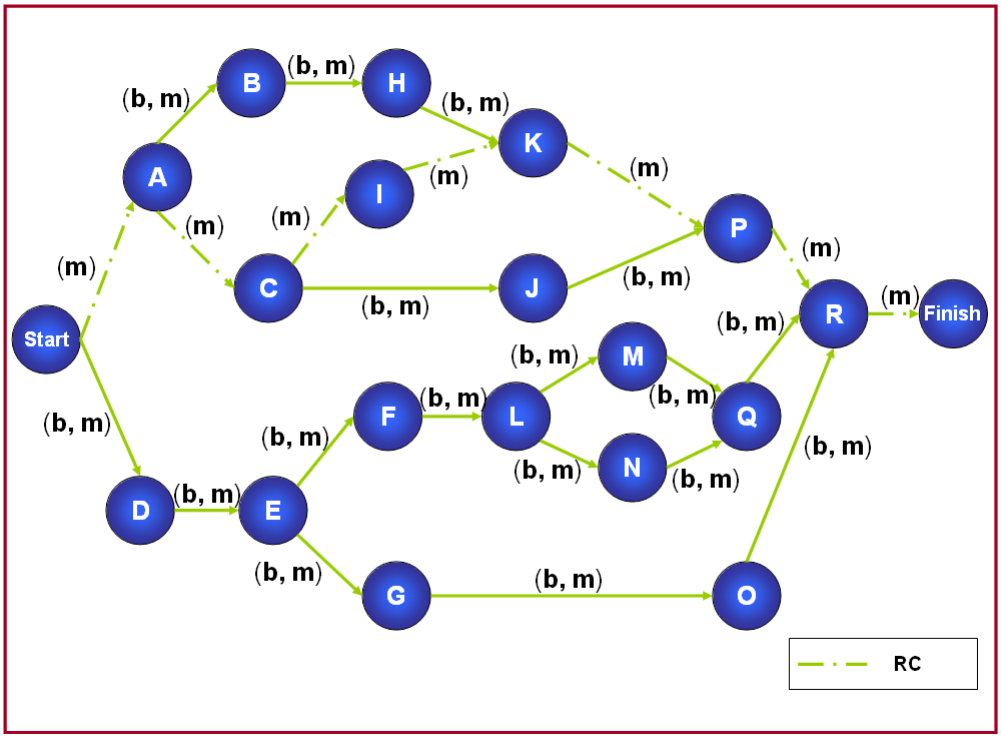

Figura 3.2: Red de actividades "extendida" mediante la inclusión de relaciones temporales

Cabe mencionar que, debido a que las relaciones temporales representadas mediante los arcos de la red se pueden ver como implícitas dentro del diagrama, en las ilustraciones restantes ya no se etiquetarán los arcos, así como se hizo en la Figura 3.2. Esto para evitar que el diagrama se oscurezca y resulte más entendible.
A continuación, se explica con mayor detalle cada uno de los elementos que componen a G.

3.1.2. Actividades

De manera intuitiva, cada actividad representa en la red un nodo con varias características (duración, holgura, tiempo temprano de inicio, etc.), el cual a su vez por sus rasgos temporales puede ser visto también como un intervalo de tiempo, mismo que se encuentra definido o acotado por los puntos (tiempos) de inicio y terminación. Ahora bien, vale la pena destacar que estos intervalos cuentan con dos rasgos distintivos: su *longitud* (duración) y su *posicionamiento* en el tiempo (es decir, en qué tiempos debe encontrarse su punto de inicio). En un caso habitual, la longitud de los intervalos (actividades) es única, puesto que en teoría ésta no debiera de variar a lo largo del proyecto. No obstante, en cuanto al posicionamiento, es necesario distinguir entre las actividades que pertenecen a la RC (ruta crítica) y las que no. Con referencia al primer caso, se puede decir que el posicionamiento también es único, puesto que $es = ls$ y es necesario—para que todo resulte a tiempo—que la actividad comience justamente en ese punto. Ahora bien, con respecto a aquellas tareas cuya holgura es diferente de cero, existen varias opciones para su posicionamiento, ya que éste puede ir desde un tiempo $t = es$ hasta $t = ls$, es decir, hasta un tiempo equivalente a $t = es + fl$. De hecho, no resulta difícil darnos cuenta que la ruta crítica es un caso especial derivado del segundo que se acaba de comentar, debido a que $fl = 0$.

3.1.3. Relaciones

Dentro del esquema propuesto, se pueden definir varios tipos de relaciones entre las diferentes actividades que conforman la red.

Primeramente, cada actividad se relaciona con sus predecesores y sucesores de manera *directa*, y esto se muestra a través de los arcos pertenecientes al grafo de actividades. Por tanto, se tienen *relaciones directas*. Asimismo, dentro de éstas, se pueden considerar adicionalmente otros tres tipos: directas entre dos actividades que no pertenecen a la ruta crítica; directas entre una actividad que pertenece a la ruta crítica y una que no pertenece; y directas entre dos actividades que pertenecen a la ruta crítica. Para fines prácticos, los primeros dos subtipos pueden considerarse como uno mismo, ya que el vector de relaciones sostenidas no cambia, mientras que entre actividades del camino crítico sí. Por consiguiente, para el primer caso, se tiene un vector con dos relaciones (b, m), y para el segundo, se tiene un vector con un único valor (m). La asignación de estos vectores se explica a continuación.

Al existir una relación en donde se involucran actividades no pertenecientes al camino

crítico, por definición se puede asumir que existe una cierta holgura en éstas (diferente de cero); sin embargo, vale la pena destacar que aún así las actividades preservan un orden establecido para su ejecución. Por ende, las dos opciones que se visualizan para llevar a cabo una actividad (viéndola desde el punto de vista de alguna de sus actividades sucesoras) son que se ejecute *antes* del comienzo de la siguiente actividad o que se ejecute de tal modo que termine justo cuando debe comenzar su sucesora (*se encuentra con*). Por ejemplo, supóngase que existe una actividad A de duración 3, cuyo tiempo temprano de inicio es $ES = x$, y cuya holgura es $FL = 2$; dicha actividad tiene como sucesora a la actividad B. Tienen cabida los dos escenarios descritos anteriormente: el primero es que A comience en el tiempo x y termine en el tiempo $x + 3$ (o en $x + 1$ para terminar en $x + 4$), y el segundo es que empiece en $x + 2$ y acabe en $x + 5$. Suponiendo que el tiempo temprano de inicio de B es $ES = x + 5$, en el primer caso tendría lugar la relación $A\ b\ B$, y en el segundo sería $A\ m\ B$. Como no se sabe exactamente la asignación de ejecuciones en el desarrollo del proyecto (en otras palabras, como ya se había visto, el posicionamiento de las actividades varía de acuerdo a fl), se conforma el vector de relaciones posibles como (b, m).

No obstante, al considerar solamente relaciones entre elementos de la ruta crítica, se sabe que éstos no tienen holgura, haciendo que su posicionamiento en el tiempo sea único. Por tanto, estrictamente hablando, todas las actividades van *seguidas* de sus predecesores y sucesores. Es por ello que el único valor factible para el vector de relaciones es (m).

Antes de proseguir, hay que recordar que las relaciones *directas* son en general *relaciones de precedencia*; por lo tanto, por la propiedad de transitividad que éstas poseen [9], dos actividades pueden precederse sin necesidad de contar con un arco que las una.

Mediante establecer las relaciones directas entre las actividades (las cuales están dadas por la constitución de la red), resulta posible deducir vectores de relaciones entre dos actividades que aparentemente no muestran conexión. A este tipo de conexiones se les puede considerar como relaciones indirectas o *inferidas*, las cuales en términos simples muestran *relaciones de concurrrencia* entre actividades y esto nos indica que se pueden realizar *paralelamente*. Para obtener estas relaciones, se introducen como datos las relaciones directas y se corre el algoritmo de inferencia descrito por Allen (el cual es implementado por la herramienta TimeGraph II). De esta forma, se sacan todas las relaciones posibles que existen entre cada par de nodos del grafo (visto de otra manera, se obtienen las relaciones con respecto al producto cartesiano del conjunto A). Sin embargo, como es de suponerse, no todas éstas van a proveer de información para el diagnóstico de discrepancias; este último subtipo—aquél que no provee información alguna—se distingue por poseer una de dos características. La primera consiste en contar con todos los elementos posibles en el vector de relaciones; esto no brinda conocimiento alguno, ya que no indica qué relaciones deben darse entre dos actividades de manera indirecta para prevenir atrasos. La segunda característica es describir relaciones de precedencia remotas entre actividades; por ejemplo, no es de mucha utilidad el saber que la primer actividad va antes que la última, a pesar de que esta información no nos es directa-

mente proporcionada por la red, como ya se había discutido. Por lo tanto, de forma intuitiva, es posible distinguir dos clases de relaciones inferidas: *relevantes* y *no relevantes*. Adicionalmente, en cuanto a las relaciones no relevantes, éstas se pueden clasificar en *de precedencia remota* y *no restrictivas*.

Por último, para propósitos informativos, cabe destacar que para todos los tipos de relaciones definidas anteriormente, existen vectores de relación *inversos*. Éstos consisten simplemente en invertir el orden de los nodos involucrados y cambiar cada relación que constituye el vector por su inverso (de acuerdo a las relaciones inversas definidas por Allen). Por ejemplo, el inverso de $(A(s)B)$ corresponde a $(B(s-)A)$.

Los tipos de relaciones recién definidos se resumen en el Cuadro 3.1

Tipo	**Subtipos**		**Ejemplos**
Directa	RC		A (m) B
	NRC		C (b, m) D
Inferida	Relevante		A (e, s, b, d) B
	Irrelevante		
		Prec. remota	C (a) D
		No restrictiva	C $(b, m, s, f, o, d, e, a, m-, s-, f-, o-, d-)$ D
Inversa			A $(sbod)$ B
			A $(siaoidi)$ B

Cuadro 3.1: Tipos de relaciones definidas para el mecanismo

3.1.4. Descripción formal

Conceptos

Proyecto

Definición: Un proyecto P se puede definir como la tupla

$$P = (F, PT, dp, G, M, PM)$$

en donde cada uno de sus elementos se lista en el Cuadro 3.2.

Nótese lo siguiente:

Elemento	Descripción
$\boldsymbol{F}$	Conjunto de todas las fechas involucradas en la ejecución del proyecto
$\boldsymbol{PT}$	Conjunto cuyos elementos son los puntos de tiempo involucrados en el proyecto
$\boldsymbol{dp}$	Duración del proyecto
$\boldsymbol{G}$	Grafo que representa la red de actividades
$\boldsymbol{M}$	Conjunto de *milestones* y $M \subset PT$
$\boldsymbol{PM}$	Conjunto de puntos de monitoreo y $PM \subseteq PT$

Cuadro 3.2: Elementos de un proyecto

- F puede visualizarse como la unión de dos conjuntos adicionales: el conjunto de las fechas que demarcan o acotan el proyecto (inicio y terminación) y el conjunto que contiene a todas las fechas intermedias. Es decir:
  - $F = FE \cup FI$, donde $FE = \{f_s, f_f\}$ y $FI = \{f_i | f_s < f_i < f_f\}$
- Todos los elementos de PT se derivan de F, de tal manera que $PT = \{p | p = \phi(f), f \in F\}$
- Análogamente, $PT = PE \cup PI$, donde $p_s = \phi(f_s)$,
  - $p_f = \phi(f_f), PE = \{p_s, p_f\}$ y
  - $PI\,\{p_i | p_s < p_i < p_f\}$

Función ϕ

En cuanto a la función ϕ, ésta tiene el propósito el convertir las fechas involucradas dentro de un proyecto a puntos de tiempo absolutos [2]. Recibe como entrada una fecha f_i y regresa un entero t_i, que representa el punto de tiempo referido por la fecha.

En general, la función trabaja de la siguiente manera:

$\phi(f_s) = 0$
$\phi(f_f) = dp$
$\phi(f_i) = t_i$, donde $0 < t_i < d$

Por ejemplo, supongamos que el inicio del proyecto es en f_s = 10-10-2006 (las fechas se encuentran en formato DD-MM-AAAA), el tiempo de terminación es f_f = 10-31-2006 y adicionalmente tenemos una fecha intermedia, que puede ser f_i = 15-10-2006. Bajo estos supuestos, se tiene que $\phi(f_s) = 0, \phi(f_f) = 21$ y $\phi(f_i) = 5$

Red de actividades (G)

Ahora bien, en cuanto a G, éste se define como $G = (A, E)$, donde A es el conjunto de tareas que se ejecutan dentro del proyecto y E define las relaciones de precedencia entre

éstas. En general, el grafo G (denominado como digrafo PERT por [10]) tiene las siguientes propiedades:

1. Existe un vértice s, llamado vértice de inicio—conocido también como el nodo *fuente*—y un vértice $f\,(\neq s)$ de terminación—conocido asimismo como nodo *sumidero* o *destino*—.
2. G no tiene circuitos dirigidos (es acíclico)
3. Cada vértice $v \in A - \{s, f\}$ se encuentra en alguna trayectoria dirigda desde s hasta f

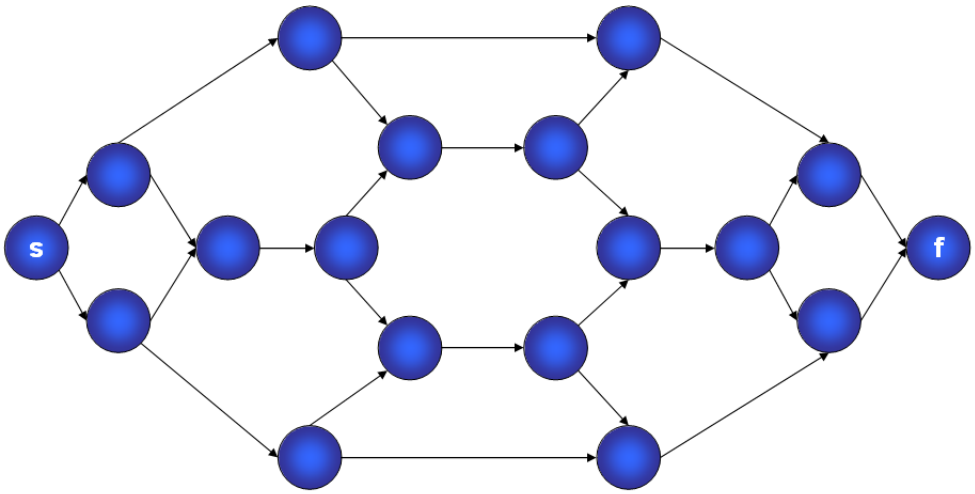

Figura 3.3: Grafo con estructura tipo PERT.

Tomando lo anterior en consideración, todas las redes de actividades que se estarán manejando siempre contarán con dos nodos "especiales", START (que corresponde al vértice s) y FINISH (que corresponde análogamente al nodo f). Esto para asegurarse que aunque haya varias actividades de inicio o de terminación, éstas se encuentren sujetas a tiempos de para comenzar y finalizar (lo cual es lógico puesto que las actividades no comienzan ni terminan en tiempos arbitrarios, sino que lo hacen dentro de los tiempos estipulados para el proyecto). De hecho, esta consideración es de suma importancia para que la máquina de RT derive inferencias correctas.

Ahora bien, en cuanto a las actividades, note—como ya se había mencionado en la sección anterior—que:

$\forall a\ a \in A \Rightarrow P\,(di)\,a$

Cada actividad perteneciente a A está constituida por una tupla con la siguiente estructura:

$$x = (id, name, state, d, es, ef, ls, lf, fl, cp, rs, rf, pr, su)$$

Cada elemento de la tupla se explica en el Cuadro 3.3

Elemento	**Descripción**
id	Etiqueta con la cual se identifica cada actividad
name	Nombre descriptivo de la actividad
state	Estado de la actividad que demarca su progreso
d	Duración
es	Tiempo temprano de inicio
ef	Tiempo temprano de terminación
ls	Tiempo tardío de inicio
lf	Tiempo tardío de terminación
fl	Holgura
cp	Pertenece o no pertenece a la RC
rs	Tiempo real de inicio
rf	Tiempo real de terminación
pr	Conjunto de actividades que preceden directamente a la actividad
su	Conjunto de actividades que suceden directamente a la actividad

Cuadro 3.3: Elementos que componen a una actividad

Donde:

- $state \in States, States = \{NS, S, F\}$ NS=no iniciada, S=iniciada, F=terminada
- $cp \in B, B = \{T, F\}$ T=Verdadero, F=Falso
- $es, ef, ls, lf, fl, rs, rf \in N$
- $pr \subset A$
- $su \subset A$

Además, nótese que:

- $cp = T \Leftrightarrow fl = 0$
- $\forall t, t \in PT$
  - $[(t < ls \ \wedge \ cp = T) \vee (t < es \ \wedge \ cp = F)] \Rightarrow state = NS$
  - $[(t \geq ls \ \wedge \ t < ef \ \wedge \ cp = T) \vee (t \geq lf \ \wedge \ cp = F)] \Rightarrow state = S$

- $[(t \geq ef \ \wedge \ cp = T) \vee (t \geq ls \ \wedge \ cp = F)] \Rightarrow state = F$
- $[(t \geq es \ \wedge \ t < ls \ \wedge \ cp = F)] \Rightarrow (state = NS \vee state = S)$
- $[(t \geq ls \ \wedge \ t < ef \ \wedge \ cp = F)] \Rightarrow (state = S \vee state = F)$

Y recordando lo anteriormente explicado:

- $ls = es + fl$
- $lf = ef + fl$

  En consecuencia,

- $fl = ls - es$ y $fl = lf - ef$

Inherentemente, cada actividad—como ya se había mencionado—cuenta con sus tiempos de inicio y terminación, así como con una duración y una cierta holgura (que en actividades de la RC equivale a cero). Del mismo modo, para propósitos de implementar la mecánica propuesta para el monitoreo del progreso en la red, se están registrando los tiempos reales de inicio y terminación, así como los predecesores y sucesores de la actividad (los cuales están dados por la constitución de la red) y el estado de progreso de la actividad. De manera adicional (sobretodo para propósitos informativos), también se incluyen en la tupla un identificador y un nombre descriptivo para cada actividad.

Ahora bien, en cuanto a las relaciones, en general cada relación y se encuentra definida por una tripleta con la estructura:

$(a1, r, a2)$, donde $(a1, a2) \in A$, y $r \subset$ Allen.

El conjunto *Allen* hace referencia a todas las relaciones temporales posibles entre dos intervalos. No obstante, conviene especificar los tipos de relaciones anteriormente mencionados, y para ello es necesario definir algunos otros conjuntos.

Para describir los valores temporales de los vectores pertenecientes a las relaciones directas, se establecen dos conjuntos adicionales: $Allen_{RC}$ y $Allen_{NRC}$. El primero contiene aquellos valores que corresponden a las relaciones directas entre actividades de la RC, mientras que el segundo abarca aquellas relaciones en las cuales al menos una actividad no pertenece a ésta.

$\text{Allen} = \{b, d, f, o, s, e, a, d-, f-, o-, s-\}$
$\text{Allen}_{\text{NRC}} = \{b, m, a, m-\}$
$\text{Allen}_{\text{RC}} = \{m, m-\}$

Nótese que $Allen_{RC} \subset Allen_{NRC}$, y que a su vez $Allen_{NRC} \subset Allen$. Vale la pena recordar que todos los tipos de relaciones a final de cuentas son subconjuntos de *Allen*. De

hecho, *Allen* podría verse como el universo en el cual se desenvuelven el resto de los conjuntos ($Allen = U$).

Otro conjunto que atañe la descripción de los tipos de relaciones establecidos es N, el cual define el vector que contiene las relaciones no relevantes. Como ya se había discutido anteriormente, estas relaciones cumplen con alguna de dos características. En el primer caso, el vector de relaciones sería igual al conjunto *Allen*. Y para el segundo, el vector sería equivalente a un conjunto x, donde $x = Allen_{NRC}$, ya que este vector nos proporciona los valores que describen a las relaciones de precedencia. En consecuencia, $N = Allen$ ó $N = Allen_{NRC}$.

Ahora bien, teniendo todo lo anterior en cuenta, es posible establecer en términos formales cada tipo de relación existente en nuestro dominio.

Relaciones directas

$r = (a1, d, a2)$, donde:

$d \in Allen_{RC}$	si relaciona sólo actividades de la RC
$d \in Allen_{NRC}$	si relaciona actividades y al menos una no pertenece a la RC

Relaciones inferidas

$r = (a1, i, a2)$, donde:

$i \subset Allen$	si es relevante
$i = Allen$	si es no relevante de tipo no restrictiva
$i \subset Allen_{NRC}$	si es no relevante de precedencia remota

Ejemplo

Suponga que se tiene un proyecto denominado "Webpage", el cual consta de realizar los componentes que conforman una página de Internet dinámica, la cual muestra parte del contenido de una base de datos. Después de realizar la descomposición de actividades y la estimación del tiempo y recursos requeridos, se ha determinado que el proyecto durará un total de 24 días, comenzando desde el primero de octubre hasta el 24 del mismo mes.

Se tiene la siguiente descomposición de actividades (WBS):

1. Diseño Back-end
   - *a*) Implementación de la BD
   - *b*) Definición de consultas a la BD
   - *c*) Programación de componentes dinámicos (conectividad con BD, servlet, etc.)
2. Diseño Front-end
   - *a*) Componentes estáticos (encabezados, pies de página, imágenes, etc.)
   - *b*) Estilo (CSS)

Para facilitar la referencia a estas actividades, se les asignarán símbolos que las representen. Comenzando de arriba a abajo, dichas actividades pasarán a ser π, σ, κ, ρ y ι, respectivamente.

Se tienen las siguientes dependencias entre actividades:

Actividad	**Dependencias**
π	–
σ	π
κ	σ
ρ	–
ι	ρ

Cuadro 3.4: Lista de dependencias para el proyecto *WebPage*

Por lo tanto, esta red corresponde a la Figura 3.4

Teniendo en cuenta los datos anteriores, se puede establecer el proyecto–junto con sus componentes–de manera formal:

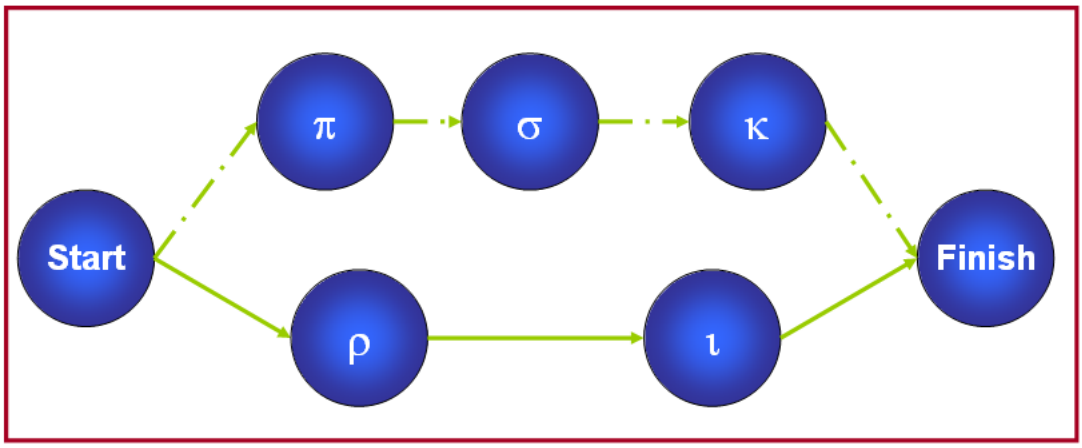

Figura 3.4: Red de trabajo para el proyecto WebPage

$$
\begin{aligned}
\text{WebPage}= \quad (\quad & \{\text{01-10-2006}, \text{02-10-2006} \ldots \text{24-10-2006}\}, \\
& \{0, 1, 2, 3, 4 \ldots 23\}, \\
& 24, \\
& (\ \{\text{Start}, \pi, \sigma, \kappa, \rho, \iota\}, \\
& \quad \{(Start, \pi), (Start, \rho), (\pi, \sigma), (\sigma, \kappa), (\rho, \iota), (\kappa, \text{Finish}), (\iota, \text{Finish})\}, \\
&) \\
& \{6, 15\}, \\
& \{0, 5, 13\} \\
) &
\end{aligned}
$$

Actividades

$\pi = (pi, \text{“Implementación BD”}, NS, 5, 0, 5, 0, 5, 0, T, -1, -1, \emptyset, \{\sigma\})$
$\sigma = (sigma, \text{“Definición consultas a BD”}, NS, 8, 5, 13, 5, 13, 0, T, -1, -1, \{\pi\}, \{\kappa\})$
$\kappa = (kappa, \text{“Componentes dinámicos”}, NS, 10, 0, 5, 0, 5, 0, T, -1, -1, \{\sigma\}, \emptyset)$
$\rho = (ro, \text{“Componentes estáticos”}, NS, 6, 0, 6, 4, 10, 4, F, -1, -1, \emptyset, \{\iota\})$
$\iota = (iota, \text{“Estilo”}, NS, 13, 6, 19, 10, 23, 4, F, -1, -1, \{\rho\}, \emptyset)$

Cuadro 3.5: Descripción de las actividades del proyecto *WebPage*

Relaciones

Directas—RC

$rd1 = (\pi, (m), \sigma)$

$rd2 = (\sigma, (m), \kappa)$

Directas—NRC

$rd3 = (\rho, (bm), \iota)$

Inferidas—Relevantes

$ri1 = (\pi, (m, b, f-, o, d-), \iota)$

$ri2 = (\pi, (e, s, s-, m, b, f-, o, d-), \rho)$

$ri3 = (\kappa, (e, s-, f-, d-, m-, f, o-, a), \iota)$

$ri4 = (\kappa, (s-, d-, m-, o-, a), \rho)$

Inferidas—Precedencia remota (No relevantes)

$ri5 = (\pi, (b), \kappa)$

Inferidas—No restrictivas (No relevantes)

$ri6 = (\sigma, (e, s, s-, m, m-, f, f-, o, o-, d, d-, a, b), \iota)$

$ri7 = (\sigma, (e, s, s-, m, m-, f, f-, o, o-, d, d-, a, b), \rho)$

Relaciones Inversas

Directas—RC

$id1 = (\sigma, (m-), \pi)$

$id2 = (\kappa, (m-), \sigma)$

Directas—NRC

$id3 = (\iota, (am-), \rho)$

Inferidas—Relevantes

$ii1 = (\iota, (m-, f, d, o-, a), \pi)$

$ii2 = (\rho, (e, s, s-, m-, f, d, o-, a), \pi)$

$ii3 = (\iota, (e, s, m, b, f-, o, f, d), \kappa)$

$ii4 = (\rho, (s, m, b, o, d), \kappa)$

Inferidas—Precedencia remota (No relevantes)

$ii5 = (\kappa, (a), \pi)$

Inferidas—No restrictivas (No relevantes)

$ii6 = (\iota, (e, s, s-, m, m-, f, f-, o, o-, d, d-, a, b), \sigma)$

$ii7 = (\rho, (e, s, s-, m, m-, f, f-, o, o-, d, d-, a, b), \sigma)$

3.2. Funcionamiento del mecanismo

Una vez teniendo en cuenta la constitución de la red—es decir, cuáles son los elementos sobre los cuales se estará trabajando, las características que poseen y las maneras diferentes en que se pueden relacionar—, se puede describir la mecánica planteada para realizar el monitoreo a las actividades y notificar en caso de detección de situaciones fuera de lo establecido como *normal*.

El principio (el cual simplemente corresponde al proceso de monitoreo definido en el Capítulo 2) que se sigue para verificar que las actividades vayan a tiempo consiste en tener o construir un *esquema ideal* (es decir, un esquema que indique lo que *debe* darse para que no existan situaciones que salgan de lo *planeado*) y contrastarlo contra el *esquema actual* o *esquema real* que a final de cuentas es el que está tomando lugar en el proyecto. Esta comparación servirá para encontrar *discrepancias*; si éstas existen, significa que se ha producido una situación en la cual las actividades no van a tiempo.

Ahora bien, siguiendo esta misma línea, debido a que al aplicar el RT sobre el grafo de trabajo se obtiene un conjunto de relaciones—de tipo relevante—posibles entre pares de actividades de acuerdo a las restricciones descritas por la red, es posible utilizar estos conjuntos de conexiones como el *esquema ideal* que servirá para asegurar que las relaciones actuales que se van dando entre las distintas actividades correspondan a lo *previsto*. Es decir, a grandes rasgos se trata de verificar que la relación que va tomando lugar entre un cierto par de actividades siempre caiga dentro del conjunto permitido, el cual se encuentra conformado o restringido por las relaciones relevantes obtenidas previamente. De esta forma, se pueden diagnosticar situaciones en las cuales exista algún atraso o adelanto. No obstante, también se toma en cuenta el caso en el que las actividades no cuentan con relaciones relevantes para construir el esquema teórico.

3.2.1. Algoritmo general

Como punto de partida, puede tomarse el conjunto de milestones del proyecto. Debido a que éstos generalmente son entregables o indican puntos de revisión, el estar monitoreando el progreso con respecto a estas fechas es algo prácticamente esencial. De igual manera, se puede suponer que el chequeo se realiza con base a la fecha más próxima de revisión; en otras palabras, se hace con respecto al punto de revisión (*milestone*) más cercano. Ése sería primer paso necesario para realizar el monitoreo: considerar las fechas importantes.

Ahora bien, ya que se cuenta con el punto de revisión más próximo, éste debe ser relacionado con alguna de las actividades de la red. Por ejemplo, si el entregable se realiza en la fecha 8 del proyecto y la actividad E termina en el tiempo 7, entonces es posible verificar el progreso comenzando por esta actividad, y de ahí hacia atrás. Para simplificar este proceso, dado que hay más de una actividad que puede relacionarse a un cierto punto de revisión y

también es probable que más de una de éstas solamente *pueda* que finalice en esta fecha (debido a la cuestión de la holgura, la cual permite que el punto de terminación de las actividades varíe), se puede seleccionar dicho nodo tomando exclusivamente en cuenta las actividades de la RC.

El siguiente paso, de manera un tanto burda, corresponde a revisar que los esquemas teóricos correspondan con los esquemas actuales. Para ello, existen dos casos básicos a considerar: 1) actividades que no cuentan con relaciones relevantes y 2) actividades que cuentan con relaciones relevantes.

Cabe mencionar que en el primer caso por lo general se habla de nodos que pertenecen a redes o subredes donde no se encuentran actividades que se puedan realizar de manera *paralela* a ellas (un ejemplo pudiera ser una red secuencial, en donde cada nodo se ejecuta antes o después que otro, pero no de manera concurrente). En estos escenarios simplemente se calcula el estado o conjuntos de estados que la actividad debería tener dado un tiempo t (esto se hace de acuerdo a las reglas anteriormente vistas), se obtiene el que actualmente tiene y se comparan. Si existe diferencia, se procede a notificar.

Siendo el segundo caso contraparte del primero, resulta sencillo ver que en este otro escenario sí existen actividades que se ejecutan de manera concurrente. Para este tipo de casos no se hace una comparación basada en los *estados*, sino más bien basada en las *relaciones*. Análogamente al caso de *actividades aisladas* (actividades sin relaciones relevantes), en este escenario se calcula—mediante la máquina de RT—el vector de relaciones posibles entre el par de actividades, se "estima" la relación actual que está ocurriendo dado el punto de monitoreo t y se verifica que ésta última forme parte del conjunto permitido de relaciones. Si existen discrepancias, es necesario notificar. Es importante considerar que el vector de relaciones posibles se modifica conforme pasa el tiempo y se van definiendo los puntos de inicio y terminación de las actividades. Por lo tanto, si durante el monitoreo no se dieron discrepancias, habría que actualizar los conjuntos de relaciones relevantes entre las actividades, puesto que, conforme pasa el tiempo, algunas de éstas ya no son posibles. De esta forma, la próxima vez que se ejecute el algoritmo, se trabajará sobre conjuntos congruentes. Ahora bien, si durante el monitoreo se dio alguna situación de atraso o adelanto, se procede a ejecutar la acción de notificación y adicionalmente esto se registra en la base de datos.

Como se puede apreciar, esta mecánica es iterativa. Dependiendo de las fechas que se establezcan para realizar los monitoreos será la cantidad de veces que se corra el proceso anteriormente mencionado. Asimismo, se importante recalcar que la descripción previa es solamente en términos generales; existen otras consideraciones que se necesitan tomar en cuenta. Para ello, a continuación se explica el algoritmo con mayor detalle.

3.2.2. Ejemplo

Para ir ilustrando el funcionamiento del mecanismo, a continuación se define—de manera un tanto intuitiva—un ejemplo.

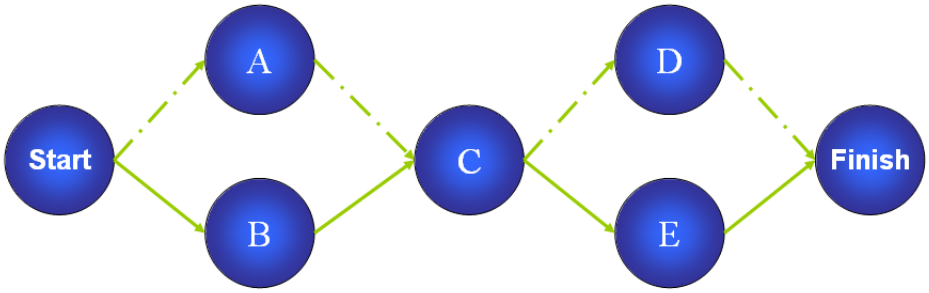

Figura 3.5: Ejemplo de red de actividades

Actividad (dur.)	**es**	**ef**	**ls**	**lf**	**Pred.**	**Suc.**
A(10)	0	10	0	10	Start	C
B(5)	0	5	5	10	Start	C
C(17)	10	27	10	27	A,B	D,E
D(18)	27	45	27	45	C	Finish
E(9)	27	36	36	45	C	Finish

El conjunto de puntos de revisión se encuentra dado por $M = \{18, 30\}$ y el conjunto de puntos de monitoreo a su vez está dado por $PM = \{7\}$

3.2.3. Algoritmo propuesto

Enseguida, se describe mediante pseudocódigo el algoritmo propuesto. Algunas de las subrutinas más importantes asimismo se listan y explican posteriormente.

Se comienza con una función de preparación, la cual revisa si el punto de tiempo en que se pretende realizar el monitoreo corresponde a los puntos establecidos para ello. Si no lo es, simplemente no se hace nada; de otra forma, se procede a hacer la tarea de monitoreo.

```
function PREPARE (p_fecha_actual) returns void
  inputs: p_fecha_actual, el punto de tiempo que representa la fecha actual

  if p_fecha_actual ∈ PM
    milestone ← GET-MILESTONE (p_fecha_actual) (I)
    milestoneActivity ← GET-MILESTONE-ACTIVITY (milestone) (II)
    RT-MONITOR (milestoneActivity, p_fecha_actual) (III)
```

Cuadro 3.6: Procedimiento PREPARE.

Obtención de la actividad relacionada con el punto de revisión (I y II)

Con el fin de obtener la actividad correspondiente al punto de revisión más cercano, primero es necesario (como ya se había planteado en la explicación general) el relacionar la fecha (punto) de monitoreo con el milestone más *cercano*. Para ello, lo que se hace es buscar aquel milestone cuya distancia con respecto a la fecha sea la menor. No obstante, hay que tomar en cuenta que sólo deben considerarse aquellos milestones que son posteriores al punto de monitoreo, pues de otra forma estaríamos contando con fechas de entregables que ya pasaron.

Lo anterior se puede ver de esta manera:

Sea $f \in PM$ y $S = \{x | x - f \geq 0 \ \wedge \ x \in M\}$, luego la función para obtener el punto de revisión más cercano se define como en el Cuadro 3.7.

Remontándonos al ejemplo planteado, tenemos que existen dos elementos pertenecientes al conjunto M: 18 y 30. Dado que $f = 7$, ambos valores pasan a formar parte de S (puesto que $18 - 7 = 11, 11 > 0$ y $30 - 7 = 23, 23 > 0$); note que en este caso, si hubiese existido un valor inferior a 7 (p.e., 5), el elemento no hubiera entrado porque simboliza un punto de

function GET-MILESTONE (f) **returns** milestone más cercano
  inputs: f, punto de monitoreo

  $milestone \leftarrow \text{argmin}_{n \in S} \{n - f\}$
  return *milestone*

Cuadro 3.7: Función para encontrar el entregable (milestone) más cercano.

tiempo que ya pasó. Por tanto, lo único que restar hacer es escoger el valor que proporcione la menor distancia hacia f, el cual representa al milestone más cercano a la fecha actual. Para el ejemplo, este punto sería 18.

Al obtener el punto de revisión más cercano, se sigue un proceso similar para encontrar la actividad relacionada, dado que ésta sería aquella cuyo tiempo de inicio es el que se encuentra a la menor distancia del milestone. Aquí también es importante notar que sólo debemos tomar en cuenta las actividades que se desarrollen *antes* de que sea la fecha del entregable.

Esto se puede ver de la siguiente manera:

Sea ES el conjunto de puntos de inicio de las actividades pertenecientes a la RC. Por otra parte, sea m el milestone más cercano el cual fue calculado previamente. Entonces, tenemos que:

$S = \{x | m - x \geq 0 \wedge x \in ES\}$

Por tanto, el procedimiento para obtener la actividad relacionada con el punto de revisión queda como se describe en el Cuadro 3.8.

function GET-MILESTONE-ACTIVITY (m) **returns** actividad relacionada al milestone
  inputs: m, milestone más cercano

  $aMilestone \leftarrow \text{argmin}_{n \in S} \{m - n\}$
  return *aMilestone*

Cuadro 3.8: Función que regresa la actividad relacionada al entregable más cercano.

De nuevo, aplicando esto al ejemplo, se tiene que $ES = 0, 10, 27$. Debido a que $18 - 27 < 0$, se sabe que la actividad con este punto de inicio (D) se encuentra posterior al milestone, y

en consecuencia no entra al conjunto S. De esta manera, los elementos de S pasan a ser solamente 0 y 10. De ellos, el que propociona la menor distancia hacia el punto de revisión es 10, por lo que la actividad relacionada a éste corresponde a C.

En el Cuadro 3.9, se describe el procedimiento principal **(III)**.

Cálculo de las relaciones relevantes (1)

El cálculo de relaciones relevantes se hace en gran parte con el paquete de Razonamiento Temporal Timegraph II [12], el cual contiene una interfaz para trabajar con intervalos de tiempo y hacer inferencias basadas en los datos que se le proporcionen.

Para nuestro ejemplo, las siguientes instrucciones se proporcionan una vez que se tiene cargado el paquete TimeGraphII [12].

```
(init-tg) ; Inicializar la red de tiempo

; Por cada relación, se hace un
;(asserta intervalo1 (relación de precedencia) intervalo2)

(asserta START (:m) A)
(asserta A (:m) C)
(asserta C (:m) D)
(asserta D (:m) FINISH)
(asserta START (:b :m) B)
(asserta B (:b :m) C)
(asserta C (:b :m) E)
(asserta E (:b :m) FINISH)

(make-tg) ; Construir la red de tiempo
(arel) ; Consultar las relaciones generadas
```

Al correr la instrucción *arel*, TimeGraph II regresa la lista de todas las relaciones existentes entre actividades. Los resultados para el ejemplo son los siguientes:

(START (:M) A)	(B (:E :S :F :D) A)	(D (:A) B)
(START (:M :B) B)	(B (:B) FINISH)	(D (:A) A)
(START (:B) FINISH)	(B (:B) E)	(E (:M :B) FINISH)
(START (:B) E)	(B (:B) D)	(E (:M- :A) C)
(START (:B) D)	(C (:M) D)	(E (:E :S :F :D) D)
(START (:B) C)	(C (:M-) A)	(E (:A) START)
(A (:M-) START)	(C (:M :B) E)	(E (:A) B)
(A (:M) C)	(C (:M- :A) B)	(E (:A) A)
(A (:E :S- :F- :D-) B)	(C (:B) FINISH)	(FINISH (:M-) D)
(A (:B) FINISH)	(C (:A) START)	(FINISH (:M- :A) E)
(A (:B) E)	(D (:M) FINISH)	(FINISH (:A) START)
(A (:B) D)	(D (:M-) C)	(FINISH (:A) C)
(B (:M :B) C)	(D (:E :S- :F- :D-) E)	(FINISH (:A) B)
(B (:M- :A) START)	(D (:A) START)	(FINISH (:A) A)

Posteriormente, se aplica un filtro que remueve aquellas relaciones producidas por nodos especiales (que no interesan para el resto del proceso, pues no generan intervalos), las relaciones de precedencia (junto con sus inversas) y las que contengan trece relaciones (lo cual nos indica que son no restrictivas). De esta forma, quedan solamente las relaciones relevantes.

Relaciones resultantes de aplicar la función FILTER (Cuadro 3.10):

```
(A (:E :S- :F- :D-) B)
(B (:E :S :F :D) A)
(D (:E :S- :F- :D-) E)
(E (:E :S :F :D) D)
```

Detección de discrepancias (2, 3 y 4)

Quizá éste es el punto clave del algoritmo planteado, y de ahí la importancia de describir este proceso con mayor detalle.

Es importante recordar que, debido a que el enfoque adoptado se basa principalmente en las relaciones *concurrentes*, existen casos que resultan difíciles de cubrir o que quizá sería más coveniente tratar de otra manera. Específicamente, se hace referencia a aquellos en los cuales no existen relaciones relevantes(referidos anteriormente como *actividades aisladas*).

El procedimiento que se optó por seguir para estos casos fue: tomar la actividad relacionada al milestone y verificar si su estado presente era el que en teoría tenía que ser; si era diferente, se examinaba la discrepancia presentada para discernir de qué tipo era: atraso o adelanto. Por otra parte, si el estado de la actividad indicaba que todavía no debería haber comenzado (NS), simplemente se mandaba llamar la rutina de monitoreo de manera recursiva. Esto último para realizar una examinación más profunda de la red y asegurar que

aunque la discrepancia se estuviera dando en actividades remotas, ésta pudiera ser detectada y notificada en caso de existir. La función encargada de llevar a cabo esta tarea (revisión de actividades aisladas) lleva por nombre CHECK-STATE.

Ahora bien, volviendo al ejemplo propuesto, vale la pena recordar que se obtuvo que la actividad C era la relacionada al punto de revisión más cercano y por ende aquella sobre la cual se realizaría el monitoreo. Debido a que—como es visible desde el gráfico que muestra la red de actividades—C no cuenta con relaciones relevantes, entra a la función CHECK-STATE para revisar su estado. Si el estado *actual* no es congruente con el que *debería* de tener, se procede a hacer una notificación.

Los valores correspondientes a las variables consideradas en la función son los siguientes: $t = 7,\ es_c = 10,\ ef_c = 27,\ ls_c = 10,\ lf = 27,\ cp = T$ y $state_c = NS$. La regla que aplica en este caso es la Regla 1, la cual apunta a que el estado actual de C debería de ser NS. Al revisar el valor de $state_c$, se comprueba que efectivamente éste pertenece al conjunto de estados permitido. Por tanto, la actividad C va a tiempo.

Si bien la función CHECK-STATE se encuentra implementada para considerar todos los casos posibles, básicamente sólo se utiliza para cubrir aquellos en los que se tienen actividades secuenciales, que no se relacionan con otras de la forma en que nos interesa. Para los casos restantes—actividades ejecutadas de manera concurrente—, se hace uso de la información proporcionada por las relaciones relevantes. Para utilizar ésta, primero se cargan dichas relaciones desde donde se hallan almacenadas, y posteriormente cada relación donde se encuentra involucrada la actividad se va revisando de manera individual.

Recordando lo anteriormente planteado, para el caso de las actividades aisladas la comparación entre los esquemas teórico y actual se realiza con base a los *estados* de las actividades, mientras que para las relacionadas ésta se hace con base—valga la redundancia—a las *relaciones*. Teniendo esto en consideración, se puede caer en la cuenta que para conocer el estado actual de una actividad, basta con extraerlo de donde se encuentra almacenado. En ese aspecto, este dato es atómico y no requiere ser calculado o estimado. No obstante, esto no sucede con las relaciones, ya que éstas se componen de intervalos que se van definiendo a lo largo del tiempo. Es decir, en un principio todos los intervalos se encuentran indefinidos (sus tiempos reales de inicio y terminación no cuentan con valores), y a medida que se va avanzando los puntos extremos comienzan a quedar establecidos. Por ejemplo, para una actividad cuyo estado es "S" tiene solamente su punto de inicio determinado.

En consecuencia, para el caso de actividades relacionadas, es necesario *calcular* o *estimar* el esquema actual, lo cual se puede lograr mediante "suponer" o "proyectar" los puntos

extremos que se encuentren indefinidos. Para saber qué puntos son los que se necesita suponer, se puede realizar una tabla en donde se vean los estados de las actividades de manera conjunta.

		B		
		NS	**S**	**F**
A	**NS**	A y B indefinidos	A indefinido B^+ indefinido	A indefinido B definido
	S	B indefinido A^+ indefinido	A^+ indefinido B^+ indefinido	B definido A^+ indefinido
	F	B indefinido A definido	A definido B^+ indefinido	A y B definidos

Ahora bien, para "suponer " un intervalo se toma $A^- = fecha{+}1$, y $A^+ = A^- {+} d$ cuando éste se encuentra indefinido (es decir, no se conocen ni rs ni rf). Si solamente se desconoce rf, entonces $A^+ = rs + d$. Cabe destacar que cuando se da el caso en que la duración real de la actividad excede a la inicialmente estipulada, a A^+ también se le agrega la cantidad de puntos de tiempo que han transcurrido desde que la actividad debiera haber terminado.

Vale la pena recalcar dos casillas de la cuadrícula anterior: la primera (NS-NS) y la última (F-F). En la primera, se cuenta con actividades que están completamente indefinidas. En este caso, lo lógico sería suponer los dos pares de puntos extremos pertenecientes a los intervalos en cuestión. Sin embargo, esto de poco sirve, ya que prácticamente se está replicando el esquema ideal (y además conlleva el hecho de que si ambas actividades se retrasan, esto no se va a poder detectar). Por lo tanto, para este caso en particular, se considera realizar una "extensión" al escenario de actividades aisladas; siendo así, solamente se revisa que sea normal el hecho de que las actividades aún no hayan comenzado (se verifica que $t \geq ls$). Si las actividades ya deberían haber arrancado, se crea la notificación; de otra manera, se mandar llamar el algoritmo de manera recursiva. Eso con respecto a la primer casilla de la tabla; ahora bien, con respecto a la última, se procede de manera similar. En lugar de solamente checar que la relación actual final pertenezca al conjunto teórico, se procede a verificar que las actividades no hayan sufrido algún adelanto al mismo tiempo. De esta manera, al contemplar estos dos casos especiales, el algoritmo alcanza a detectar casos en los que—si se restringe solamente a la comparación entre esquemas—el mecanismo podría exhibir fallas (parte de esto se verá durante el desarrollo de las pruebas).

Una vez suponiendo los puntos que hagan falta, se obtiene la relación que ocurre si los puntos calculados se dan en la vida real. Para ello, se utilizan las definiciones de las relaciones temporales con base a los puntos de inicio y fin [1].

Las relaciones $a, o-, m-, d-, s-,$ y $f-$ se pueden derivar de la tabla anterior, tomando en cuenta que ahora B es A y viceversa.

Ahora bien, para ilustrar el uso de la mecánica planteada en esta parte, se recurrirá de nueva cuenta al ejemplo estipulado en la figura (aquí podemos poner referencia). Como se podrá recordar, debido a que aún no era tiempo de que la actividad C comenzara, el algoritmo se mandará llamar de manera recursiva, tomando para su ejecución a los predecesores de C (A y B). Estos nodos sostienen relaciones relevantes entre ellos mismos, puesto que $(A\ (e, s-, f-, d-)\ B)$ y por ende $(B\ (e, s, d, f)\ A)$. Supongamos que el mecanismo revisa primero A; más tarde, regresará a revisar B.

Dado que tanto A como B ya han comenzado, les corresponde la quinta casilla (leyendo de arriba hacia abajo y de izquierda a derecha) de la tabla (referencia). Por ende, se supondrán los puntos de terminación de ambas actividades; siendo que la duración de A es de 10, $A+^{=}10$ y siendo que la duración de B corresponde a 5, $B^{+} = 6$. Una vez teniendo todos los puntos, es necesario construir la relación actual. Puesto que $A^{-} = B^{-} \wedge B^{+} < A^{+}$, la relación actual pertinente es $(A\ (d-)\ B)$ (es decir, A *contiene* a B).

Como ya se había estipulado, lo siguiente es ver si esta relación se encuentra dentro del conjunto permitido. Puesto que $d- \in \{e, s-, f-, d-\}$, se detecta que no existen discrepancias.

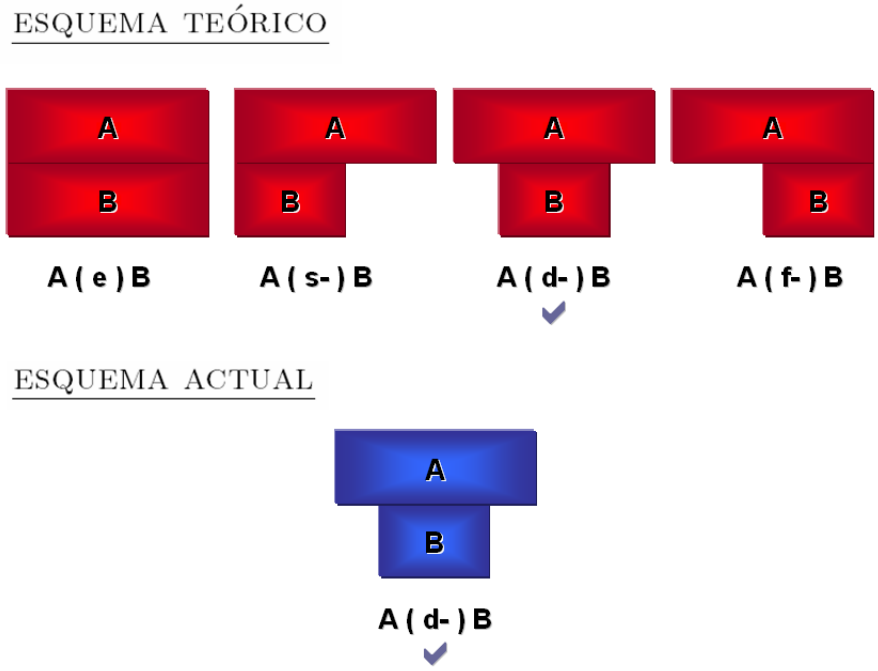

Figura 3.6: Comparación de esquema actual vs. esquema teórico

No obstante, supóngase el caso en el cual, dado $t = 7$, $B^{-} = 6$. Haciendo los cálculos apropiados, daría que $B^{+} = 11$, y por tanto la relación entre A y B sería $(A(o)B)$, la cual

no se encuentra en el vector de relaciones demarcado como ideal. Esta situación claramente marca un retraso, ya que B no alcanza a terminar en la fecha estipulada como su punto de terminación tardío.

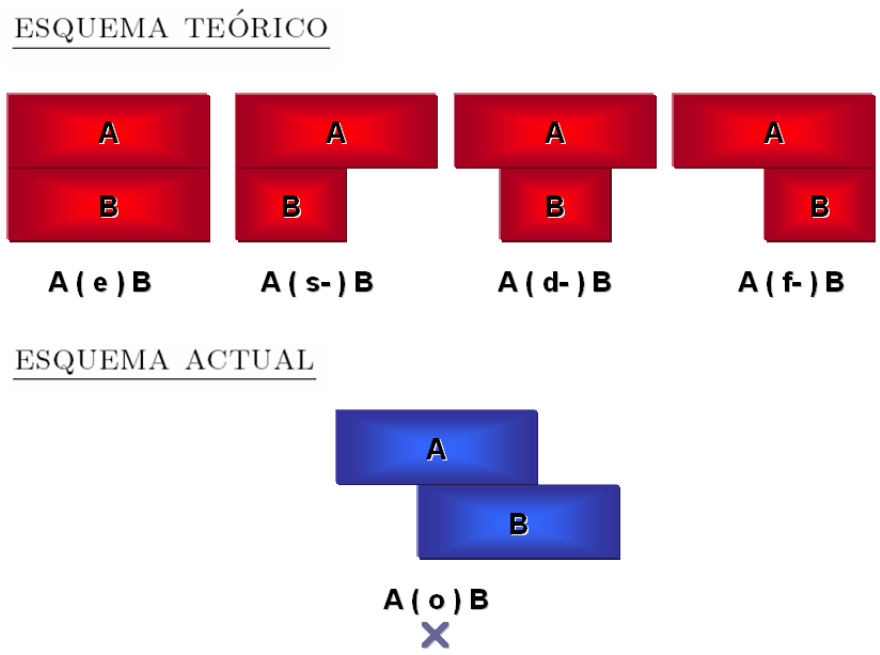

Figura 3.7: Comparación de esquemas.

Cabe destacar que, una vez que se ha revisado la relación (X (vector) Y), el algoritmo recuerda esta relación, de tal suerte que al presentarse (Y (vector inverso) X), ésta ya no se revisa. Lo anterior es para validar que no se repasen relaciones que resulten redundantes, con el propósito de ahorrar tiempo y esfuerzo computacional. Por tanto, viendo una vez más el ejemplo, al llegar a la inspección de la actividad B, la mecánica simplemente saltará esta relación.

```
function RT-MONITOR (activity, t) returns void
  inputs: activity, actividad respecto a la cual se realiza el monitoreo
          t, punto de tiempo en el que se ejecuta el monitoreo

  Relevant ← conjunto de relaciones relevantes de activity (1)

  if Relevant = ∅ //No existen rel. relevantes para activity
    Comparar estado actual de activity vs. estado que debería tener dado t (2)
    if existen discrepancias entre estados
       DO-NOTIFICATION({activity})
    else if no existen discrepancias, pero activity no ha comenzado
       Predecesors ← actividades predecesoras de activity
       for each pred ∈ Predecesors
          RT-MONITOR (pred, t) // Revisar cada predecesor recursivamente

  else // Sí existen relaciones relevantes
    for each relation ∈ Relevant
       // Con base en el estado del par de actividades de la relación,
       //  tomar curso de acción (3)
      if  actividades no han comenzado
        revisar si las actividades ya deberían haber comenzado
        if actividades ya deberían haber comenzado
          DO-NOTIFICATION({activity, relact})
        else
          Predecesors ← predecesores de activity
          for each pred ∈ Predecesors
            RT-MONITOR(pred, t)
      else if alguna de las actividades ha comenzado
           actual ← cálculo de la relación actual entre las actividades (4)
          if actual ∈ Relation
            relation ← relación con vector de relaciones actualizado (5)
          else
            DO-NOTIFICATION({activity, relact}) (6)
      else
        if alguna de las actividades está adelantada
          DO-NOTIFICATION ({activity, relact})
```

Cuadro 3.9: Procedimiento principal del mecanismo de monitoreo.

function FILTER (*Inferred*) **returns** *Relevant*
  inputs: *Inferred*, conjunto de relaciones inferidas

  for *inferred_rel* $\in$ *Inferred*
  relations_vector $\leftarrow$ vector de relaciones inferido
  if *relations_vector* $\neq$ *Allen* $\wedge$
    relations_vector $\neq \{b, m\} \wedge$
    relations_vector $\neq \{m-, a\}$
     relevant_rel $\leftarrow$ *inferred_rel*

  if relación no es inversa a ninguna otra
    Relevant $\leftarrow$ *Relevant* $\cup \{$*relevant_rel*$\}$

Cuadro 3.10: Función para filtrar relaciones relevantes.

function CHECK-STATE (*activity, t*) **returns** si la actividad está a tiempo o no
  inputs: *activity*, actividad a revisar
  t, punto de tiempo en que se realiza el monitoreo

  $es_a \leftarrow$ `es` de *activity*
  $ef_a \leftarrow$ `ef` de *activity*
  $ls_a \leftarrow$ `ls` de *activity*
  $lf_a \leftarrow$ `lf` de *activity*
  $cp_a \leftarrow$ `cp` de *activity*
  $state_a \leftarrow$ estado actual de *activity*

  if $(t < ls \ \wedge \ cp = T) \vee (t < es \ \wedge \ cp = F)$ `//Regla 1`
    State $\leftarrow$ { `NS` }
  else if $(t \geq ls \ \wedge \ t < ef \ \wedge \ cp = T) \vee (t \geq lf \ \wedge \ cp = F)$ `//Regla 2`
    State $\leftarrow$ { `S` }
  else if $(t \geq ef \ \wedge \ cp = T) \vee (t \geq ls \ \wedge \ cp = F)$ `//Regla 3`
    State $\leftarrow$ { `F` }
  else if $(t \geq es \ \wedge \ t < ls \ \wedge \ cp = F)$ `//Regla 4`
    State $\leftarrow$ { `NS` , `S` }
  else if $(t \geq ls \ \wedge \ t < ef \ \wedge \ cp = F)$ `//Regla 5`
    State $\leftarrow$ { `S` , `F` }

  if $state_a \in$ State
   return true
  else
   return false

Cuadro 3.11: Función que revisa si el estado actual de una actividad corresponde al que debería de presentar.

		B		
		NS	**S**	**F**
A	**NS**	Verificar si las actividades ya deberían haber comenzado. Si no deberían, verificar recursivamente.	$A^- < B$. Suponer B^+, A^- y A^+	Sólo a es posible
	S	$A^- < B^-$. Suponer B^-, B^+ y A^+	Suponer A^+ y B^+	Suponer A^+
	F	Única relación posible es a	Suponer B^+	Las actividades ya se terminaron. Verificar si hubo adelanto.

Cuadro 3.12: Acciones a tomar dependiendo del estado de las actividades.

Relación	**Construcción con base en puntos de inicio y fin**
A b B	$A^+ < B^-$
A m B	$A^+ = B^-$
A o B	$(A^- < B^-) \ \wedge \ (A^+ < B^+) \wedge (A^+ > B^-)$
A e B	$(A^- = B^-) \ \wedge \ (A^+ = B^+)$
A d B	$(A^- = B^-) \ \wedge \ (A^+ < B^+)$
A s B	$(A^- = B^-) \ \wedge \ (A^+ < B^+)$
A f B	$(A^- > B^-) \ \wedge \ (A^+ = B^+)$

Cuadro 3.13: Definición de relaciones temporales con base en los puntos extremos de los intervalos.

Actualización de las relaciones (5)

Debido a que conforme transcurre el tiempo las actividades se van definiendo, las relaciones posibles entre ellas cada vez se hacen menos. Por ende, estos cambios deben verse reflejados en el conjunto de relaciones relevantes; de otra manera, el algoritmo no podría funcionar correctamente, ya que estaría admitiendo posibilidades que ya no son factibles en una situación donde no existen discrepancias. Para ello, resulta necesario ir actualizando el conjunto de relaciones relevantes. Asimismo, este proceso sirve validar que siempre se tengan relaciones viables (recordemos que el RT solamente sabe el orden en que van las actividades y es lo único que toma para hacer las inferencias, por tanto, pudiera ser que en casos específicos arrojara relaciones que por cuestiones de duración u otras características no sean factibles).

Esta actualización se hace por cada ocasión en que se ejecuta el algoritmo, y se realiza si no se presentaron discrepancias. De otra forma, no se procede a refrescar las relaciones relevantes, ya que el par de actividades (y quizá otros nodos fuera de éstos dos) probablemente tendrá que ser modificado para reflejar las consecuencias del atraso o adelanto. Asimismo, la actualización sólo se hace cuando al menos una de las actividades tiene estado diferente a NS; el caso contrario nos indica que las actividades aún no se encuentran definidas y puede ser que cualquiera de las relaciones planteadas por el RT llegue a darse.

Si bien este proceso pudo haber sido ejecutado de varias maneras, en este caso se optó por utilizar el concepto de *semi-intervalos*, ya que éstos presentan la idea de información incompleta, y así es justamente como se encuentran las actividades que aún no llegan a ser completadas: sus respectivos intervalos no se encuentran totalmente definidos. Siendo esto algo importante y que debemos tener en mente, resulta atractivo utilizar la idea planteada por Freksa [11].

Por lo tanto, una vez teniendo la relación de semi-intervalos correspondiente, es cuestión de empatarla con las relaciones relevantes y dejar sólo aquellas que ambos conjuntos tengan en común. Viéndolo de manera más formal, sea *Relations* el conjunto de relaciones relevantes de entre dos actividades y *SemiIntervals* el conjunto de relaciones obtenidas mediante aplicar el concepto de semi-intervalos a estas dos actividades. Por tanto, $Relation = Relation \cap SemiIntervals$. En vista de lo establecido con anterioridad, es posible armar la función de actualización.

La actualización mediante semi-intervalos se explica con mayor detalle en el Cuadro 3.15.

Siguiendo con el ejemplo, debido a que (con los datos proporcionados hasta la fecha) las actividades A y B no presentaron discrepancias, se procede a refrescar el vector de relaciones (vale la pena destacar que la actualización se realiza tanto para la relación A-B, como para la relación B-A). Para ello, lo primero es detectar el semi-intervalo que ocurre entre ambos nodos; debido a que $state_A = S$, $state_B = S$, y además $A^- < B^-$, éste corresponde a A *oc* B.

function UPDATE (*activity, relact, Relation*) **returns** vector de relaciones actualizado
inputs: *activity* y *relact*, actividades involucradas en la relación
Relation, vector de relaciones actual

SemiIntervals ← GET-SEMI-INTERVALS (*activity, relact*)
NewRelation ← *Relation* ∩ *SemiIntervals*
return *NewRelation*

Cuadro 3.14: Función que actualiza los vectores de relaciones existentes entre intervalos.

	NS	**S**	**F**
NS	N/A	A *yo* B	A b B
S	A *ol* B	(*)	(*) ∩ A *sv* B
F	A *a* B	(*) ∩ A *sb* B	Relación real

Donde:

(*)=
Si $A^- < B^-$, entonces A *oc* B
Si $A^- = B^-$, entonces A *hh* B
Si $A^- > B^-$, entonces A *yo* B

A *oc* B = $\{o, f-, d-\}$
A *hh* B = $\{s-, e, s\}$
A *yc* B = $\{d, f-, o-\}$

A *oc* B ∩ A *sb* B = o
A *hh* B ∩ A *sb* B = s
A *yc* B ∩ A *sb* B = d
A *oc* B ∩ A *sv* B = $d-$
A *hh* B ∩ A *sv* B = $s-$
A *yc* B ∩ A *sv* B = $o-$

Cuadro 3.15: Semi-intervalo correspondiente, de acuerdo a los estados de las actividades.

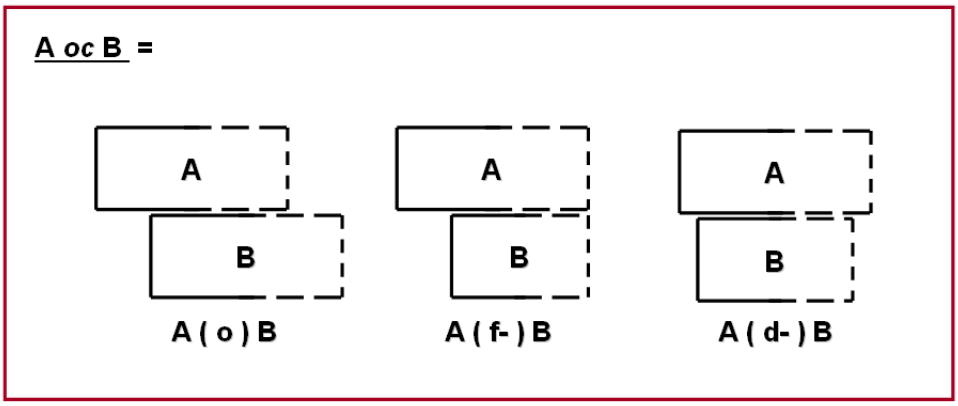

Figura 3.8: Interpretación de semi-intervalos en las actividades

Asimismo, como oc equivale al vector $(o, f-, d-)$, entonces $oc \cap (e, s-, f-, d-) = (f-, d-)$. Por lo tanto, la relación actualizada es $A(f-, d-)B$.

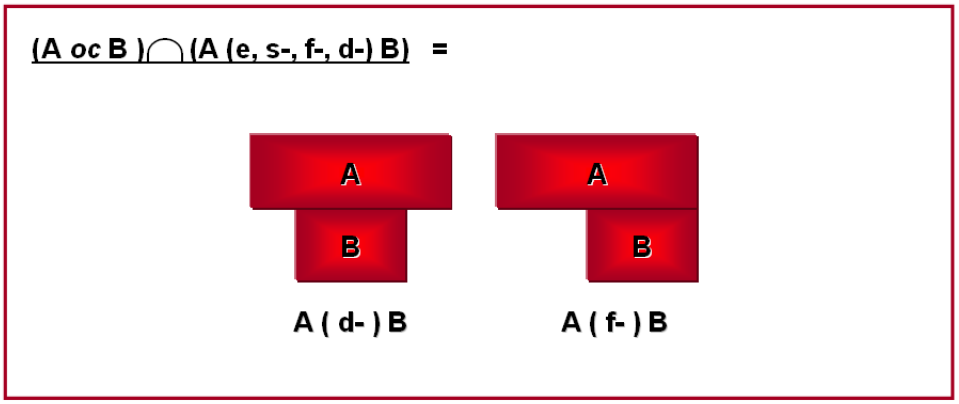

Figura 3.9: Proceso de actualización de las relaciones temporales

Notificación (6)

Una vez que se detecta una discrepancia, ésta se debe notificar a los involucrados para que se hagan los ajustes necesarios. En nuestro caso, consideramos tres principales involucrados a quienes se debe hacer la notificación: los agentes encargados de la actividad, los agentes encargados de las actividades *sucesoras* (sólo consideramos a las que *directamente* suceden, es decir, los "hijos" del nodo en la red) y los agentes encargados de las actividades *concurrentes* al nodo en cuestión. Los primeros, por cuestiones lógicas, deben enterarse de la discrepancia ocurrida; a los sucesores de la actividad también es necesario que se les notifique, puesto que

function DO-NOTIFICATION (*Activities*) **returns** `void`
  inputs: *Activities*, conjunto de actividades involucradas en discrepancia

  `//Revisar si se trata de un atraso o un adelanto`
  delay_type ← tipo de discrepancia (atraso, adelanto)

  Direct ← encargados de act. sucesoras
  Parallel ← encargados de actividades paralelas
  Who ← *Direct* ∪ *Parallel*

  CREATE-NOTIFICATION (delay_type, Who)

Cuadro 3.16: Procedimiento de Notificación

a estas actividades les afecta en su tiempo de ejecución el hecho de que sus predecesores se atrasen o adelanten; y, por último, también sería conveniente hacer la notificación porque es posible que sean asimismo afectados por la discrepancia encontrada (por ejemplo, si una actividad se adelanta, puede ser que la cadena a la que pertenecía deje de ser la ruta crítica, haciendo con esto que otro conjunto de nodos ahora la conforme, o pudiese ser que si una actividad se atrasa se brinde una mayor holgura a las que paralelamente se estaban ejecutando). El procedimiento de notificación se ilustra en el Cuadro 3.16.

Debido a la naturaleza del mecanismo, resulta posible su integración a sistemas como JITIK. El esquema de notificación utilizando este sistema se discute a continuación. Asimismo, se brinda una breve descripción de JITIK.

Notificación a través del sistema JITIK

A medida que las organizaciones crecen y, tanto la información como el conocimiento cada vez se vuelven más importantes para adquirir ventajas competitivas, es que surge la necesidad de crear una cultura basada en conocimiento. Por tanto, es conveniente tener un flujo efectivo de información, sobretodo en casos donde cada día hay más fuentes y se vuelve más complejo su análisis. Ésta es la principal motivación detrás del proyecto JITIK (*Just-In-Time Information and Knowledge*). Su objetivo central es hacer llegar la *información correcta* a la *persona correcta*—las cual se caracteriza por la división o área dentro de la organización, su nivel o responsabilidad, o sus intereses—en el *tiempo correcto* [4], de tal forma que dada una situación específica, se pueda actuar mejor. Para ello, JITIK se vale del uso de diferentes tecnologías de información, pero especialmente se basa en *sistemas multiagente* [14].

Como ya se vio previamente, la labor del mecanismo de monitoreo planteado termina al detectar que no existieron discrepancias de origen temporal en las actividades, o al detectar

que sí las hubo, para lo cual solamente *genera* un aviso donde se describe la situación. Partiendo de estos resultados—los cuales se pueden ver como las entradas al proceso de control—se vuelve importante decidir qué hacer con tales notificaciones, ya que se requiere actuar cuando sucede algo fuera de lo planeado. Como ya se había planteado en la sección anterior, una acción posible es *distribuir* esta *información* a los *interesados*. En ese aspecto, la solución que ofrece JITIK se pueden alinear a las necesidades de control del proyecto. Por tanto, se puede integrar el mecanismo de monitoreo a este sistema.

Ahora bien, de nuevo cabe hacer la pregunta: *¿cómo puede llevarse a cabo la integración?* Para ello, cabe destacar que dentro del esquema de JITIK existen varios tipos de agentes, principalmente tres: el agente de sitio (*SiteAgent*), el agente personal (*PersonalAgent*) y el agente de notificación (*NotificationAgent*). A grandes rasgos, el agente de sitio recibe todos los mensajes que se generan y los redirige ya sea a los agentes personales o de notificación (dependiendo si el usuario cuenta con agentes asignados); el agente personal lleva el mensaje al usuario con el que se encuentra relacionado, y el de notificación se encarga de transmitirlo a aquellos usuarios que no tienen agentes personales. Por lo tanto, para manejar la notificación que se registra a partir de una discrepancia encontrada, el esquema podría consistir en enviar un mensaje al agente de sitio, y éste a su vez se encargaría de distribuirlo a los agentes personales de los usuarios involucrados y agentes de notificación (para usuarios que por alguna razón no cuenten con personales). Al pasarles esta información, los agentes harán la labor que tengan establecida.

De esta forma, las notificaciones se pueden realizar utilizando la plataforma multiagente que brinda JITIK.

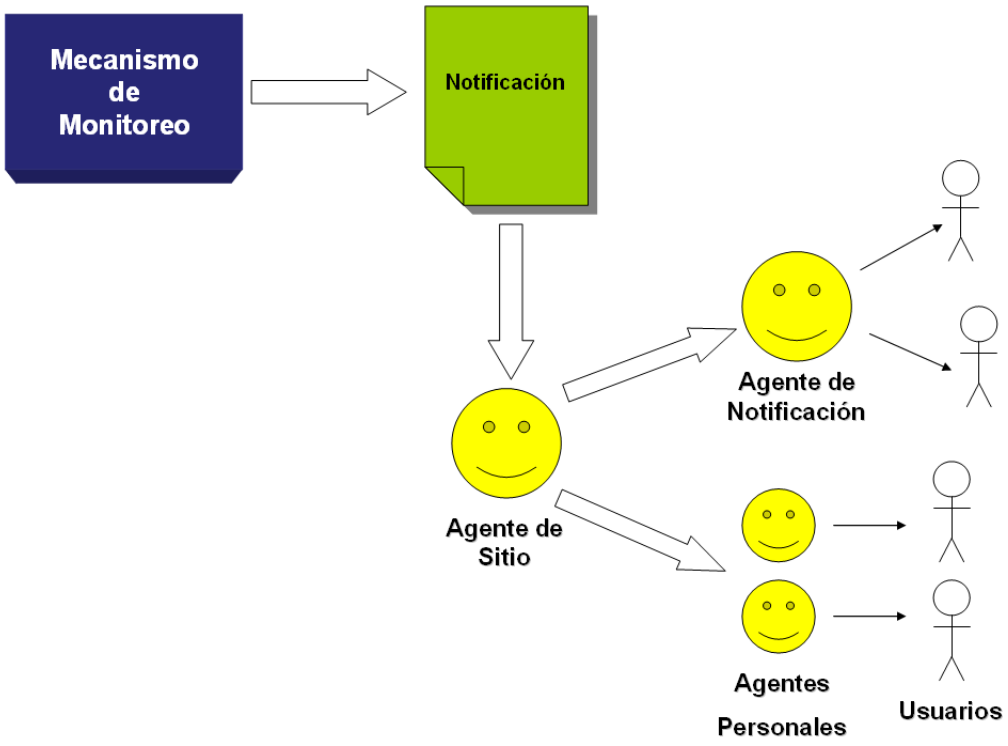

Figura 3.10: Esquema de notificación en el sistema JITIK

3.2.4. Prototipo

Para ver el funcionamiento del algoritmo y realizar pruebas sobre él, se trabajó en un prototipo que implementara lo descrito anteriormente (tratando principalmente aquellos puntos relacionados con el uso del RT). A continuación se muestra el proceso general que sigue el prototipo.

En una base de datos se guardan los datos más importantes de la red: las características de los nodos (es, ls, pertenencia a la ruta crítica) y las relaciones que constituyen los arcos del grafo. Una vez contando con esta información, es posible correr un proceso que tome como entrada el *flujo* de la red y lo convierta a primitivas entendidas por TimeGraph II, las cuales expresen las relaciones temporales que se necesitan para realizar las inferencias.

Asimismo, cuando ya se ha extraído esta información de la BD, se corre otro proceso para invocar a TimeGraph II y pasarle las relaciones temporales, de tal modo que con esos datos pueda hacer las derivaciones correspondientes entre todos los pares de actividades. Esta información se consulta y se registra para posteriormente filtrarla con el fin de solamente obtener las relaciones relevantes.

Las relaciones relevantes adquiridas se almacenan de nuevo en la base de datos, y esto da pie para que ya se pueda ejecutar propiamente la tarea de monitoreo. Esta última se realiza tal como se ha descrito anteriormente. Al detectar atrasos o adelantos, se elabora una notificación, misma que se guarda en la base de datos y se obtienen los agentes a los cuales se debe mandar ésta. En caso contrario, se hacen modificaciones a los registros pertenecientes a la tabla de relaciones relevantes para reflejar las actualizaciones.

La implementación del prototipo se encuentra realizada en dos lenguajes de programación: LISP (Allegro Common LISP) y Java (JDK versión 1.4.2–09), debido a varios factores. En primer lugar, al encontrarse TimeGraph II codificado en ACL, resulta necesario trabajar con este lenguaje, puesto que se necesita manipular la herramienta de tal manera que reciba los datos de las redes de actividades y asimismo arroje los resultados obtenidos en algún medio de almacenamiento. Por otra parte, se utiliza Java debido a que el trabajar con un lenguaje de paradigma imperativo provee mayores facilidades para realizar ciertas tareas, tales como la comunicación con la base de datos y asimismo permite realizar extensiones significativas a lo ya construido.

Para construir el "puente" entre los dos lenguajes, se hace uso tanto de archivos de texto como de una BD relacional (MYSQL) y además de comandos de Java que permiten lanzar aplicaciones ejecutables (aprovechando para ello la ventaja que nos da ACL al permitirnos crear ejecutables a base de programas hechos en LISP).

Si bien durante el presente capítulo se ha descrito de manera general el prototipo, a lo largo del siguiente se hará aún mención de algunas de sus funcionalidades y se podrá ver un

poco más a detalle cómo opera.

Capítulo 4

Pruebas

Con el fin de revisar su funcionamiento y de ver si cumple con su propósito, es necesario ejecutar pruebas sobre el mecanismo planteado en el capítulo anterior. Para ello, se armaron varios escenarios, y de éstos se derivaron casos de prueba específicos. La descripción de estos escenarios y los resultados obtenidos por el mecanismo se describen a lo largo del presente capítulo.

4.1. Selección de escenarios de prueba

Si bien existen innumerables redes de trabajo que podrían usarse para hacer experimentos, con el fin de poder brindar un panorama general y lo más completo posible, se decidió seleccionar tres esquemas (escenarios) sobre los cuales se ejecutarían las pruebas: 1) red secuencial, 2) red estrictamente paralela y 3) red concurrente (mixta). A continuación se explica por qué se hizo de esta manera.

En las siguientes secciones, se plantea visualizar a la red de actividades de dos maneras: 1) como estructura de datos Y 2)con base al tipo de relaciones que poseen (enfocado a razonamiento). Esto, con el fin de comprender las clasificaciones y ver que las redes de actividades embonan en los tipos que cada clasificación contiene.

4.1.1. Clasificación con base a la topología del grafo

Viendo los componentes del grafo, es posible crear una clasificación que va de acuerdo a la constitución "física" de la red. Para ello, convendría ver a la red de actividades como una estructura de datos. Siguiendo esta premisa, se sabe que por definición el grafo $G = (A, E)$ que compone a la red, los cuales dentro del contexto de proyectos no tienen duración, pero son relevantes porque establecen el principio y el final de la ejecución: nodo de inicio o fuente y nodo de terminación o sumidero (denominados en la presente sección como S y F, respectivamente). Asimismo, por definición, se ha establecido que estos nodos tienen restricciones sobre sus grados de entrada y salida, ya que $ge(S) = 0$ y $gs(F) = 0$. Tomando esto en cuenta y haciendo uso precisamente de los grados de entrada y salida [13], es posible armar los tipos de redes anteriormente mencionados.

Redes secuenciales

Descripción.- Es aquella en la que todas las actividades forman una sola cadena que sigue una secuencia (y por ende se tienen exclusivamente relaciones de precedencia entre los nodos).

Características.- La red de tipo secuencial cumple con las siguientes características (adicionales a las ya impuestas por la definición de este tipo de grafo):

1. $gs(S) = 1$, [El grado de salida de S es exactamente 1]
2. $ge(F) = g(S) = 1$ [El grado de salida de F—que es igual al de S—es también de uno]
3. $\forall n\ n \in A,\ ge(n) = 1 \wedge gs(n) = 1$ [El grado de entrada cualquier otro nodo es exactamente 1 y el grado de salida es exactamente 1]

En este sentido, una red vacía (con solamente nodos de inicio y terminación) es de tipo secuencial, una red con un solo nodo también es de tipo secuencial, y en general todas aquellas que cumplan con estas restricciones.

Ahora bien, estas redes pueden constar de un número arbitrario de nodos (2 en delante si contamos el fuente y el sumidero), pero se pueden reducir a redes más pequeñas por la propiedad de transitividad que existe en las relaciones de precedencia [9], lo cual se hace mediante *englobar* actividades que tienen relaciones de precedencia. De hecho, viendo los nodos como intervalos de tiempo, se puede caer en la cuenta que es siempre posible–debido a que la única secuencia de nodos en el grafo es la ruta crítica–juntar dos intervalos que van seguidos (es decir, que tienen relación m) en un súper-intervalo cuyo punto de inicio es el punto de inicio de la primer actividad y su punto de terminación es el punto de terminación de la penúltima actividad. (ver ilustración)Es de esta forma que al probar el mecanismo con una red sencilla de dos nodos, se pueden extender los resultados a redes—de este mismo tipo—de mayor tamaño.

Redes estrictamente paralelas

Descripción.- Esta clase de redes es aquella en la que solamente se tienen actividades que se ejecutan de manera concurrente (el único flujo es el que se da entre los nodos S, F y cada una de las actividades por separado). Son redes en las que se tiene más de una cadena conformada por una sola actividad.

Características.- La red de actividades estrictamente concurrente cuenta con las siguientes características:

1. $gs(S) > 1$ [El grado de salida de S es mayor a 1]

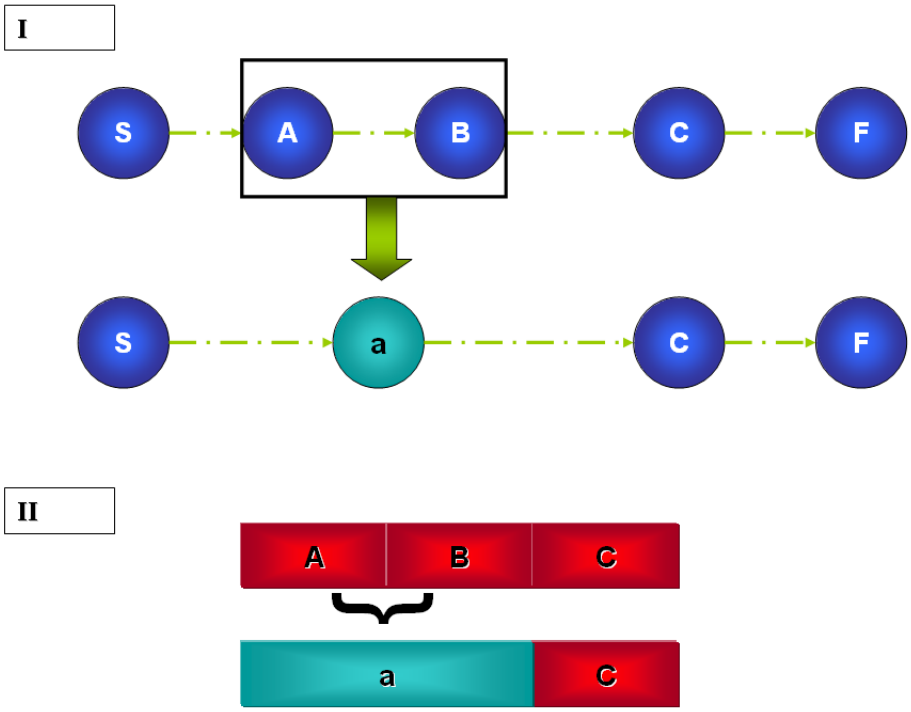

Figura 4.1: Agrupación de actividades en redes secuenciales (I) y su equivalencia desde la perspectiva de intervalos de tiempo

2. $ge(F) = g(S)$ [El grado de entrada de F es igual al de salida de S, puesto que los arcos de todas las actividades están relacionados con estos dos nodos]

3. $\forall n \; n \in A, ge(n) = 1 \wedge gs(n) = 1$ [El grado de entrada de cualquier otra actividad es exactamente uno, al igual que su grado de salida. Asimismo, el arco incidente a n proviene de S y el arco incidente desde n va dirigido hacia F]

Cabe destacar que la diferencia más importante entre los dos tipos de redes antes mencionados es el grado de S. Básicamente esto es lo que define a ambas clases. Asimismo, nótese también que mientras que en las redes secuenciales solamente existen relaciones de *precedencia*, en las redes estrictamente paralelas se dan exclusivamente relaciones de *concurrencia* entre nodos no especiales.

De manera similar a las redes secuenciales, las redes estrictamente concurrentes también pueden contar con un número arbitrario de nodos (4 en delante, contando la fuente y el sumidero), el cual se acrecienta a medida que se conecta un nuevo nodo a S y F. Ante esto, surge la cuestión de si se pueden reducir estas redes de tal manera que una red con X cantidad de nodos se pueda ver como el caso más pequeño.

Debido a que—por definición—la red de actividades cuenta con una ruta crítica, siempre habrá una actividad (de hecho, en redes estrictamente concurrentes tiene que ser exactamente una) que pertenecerá a ésta. Por lo tanto, todas las relaciones relevantes estarán conectadas a dicho nodo del grafo. ¿Cómo podemos saber esto? Básicamente es una cuestión de lógica: cada actividad se encuentra restringida en su posicionamiento con respecto a la ruta crítica. De hecho, este tipo de red en específico cuenta con un solo vector de relaciones posibles, que es $(e, s-, f-, d-)$ (y su inverso), ya que—en teoría—la actividad perteneciente a la RC contiene a todas las demás (viéndolo de manera complementaria, todas las otras actividades se deben realizar "durante" la ejecución de ésta). De igual manera, las actividades no pertenecientes a la RC no se encuentran restringidas entre ellas mismas (por lo que el RT siempre arrojará el vector que contiene todas las relaciones de Allen) debido a que se pueden ejecutar en un orden totalmente arbitrario.

En consecuencia, cualquier red estrictamente paralela se puede descomponer en subredes que pueden tratarse de manera "independiente", puesto que lo único que importa en este caso es la relación de cada actividad con respecto a la ruta crítica (ver ejemplo).

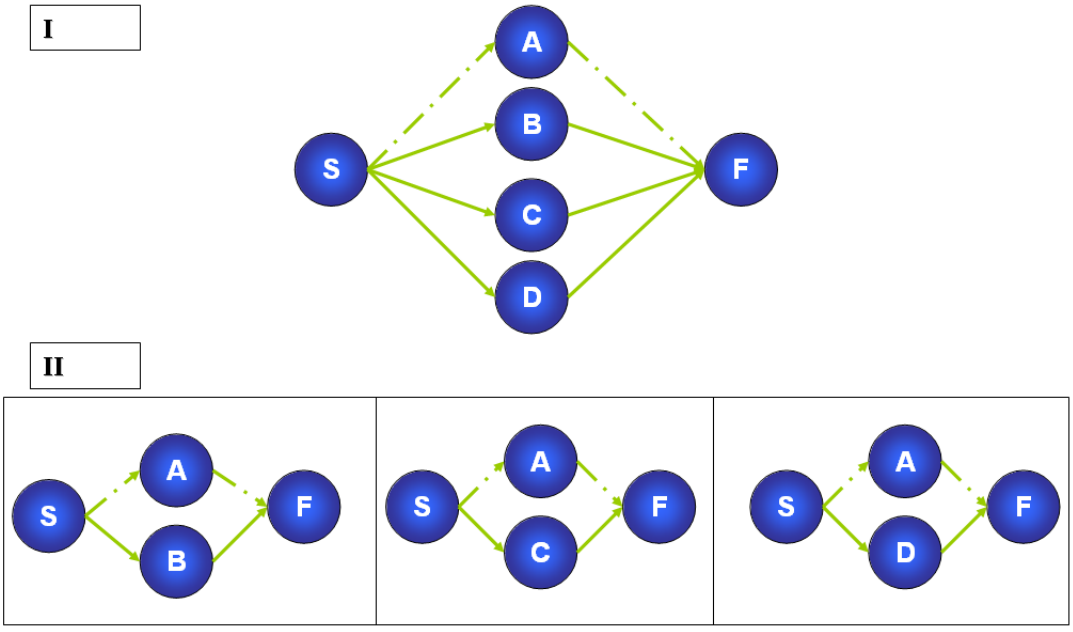

Figura 4.2: Independencia en redes estrictamente paralelas

Redes concurrentes (mixtas)

Definición.- Son aquellas redes en las cuales se tienen al menos dos secuencias (cadenas) de actividades, y al menos una de éstas contiene dos ó más nodos.

Teniendo en cuenta lo anterior, podemos decir que en la red mixta cuenta con las siguientes características:

1. $\exists n\ n \in A \Rightarrow ge(n) > 1$ [Al menos uno de los nodos es conectado por dos ó más nodos]
2. $\exists n\ n \in A \Rightarrow gs(n) > 1$ [Al menos uno de los nodos incide dos ó más nodos]
3. $\exists e1, e2\ e1 \in E \wedge e2 \in E \wedge e1 = (u, v) \wedge e2 = (v, x) \Rightarrow u \neq S \wedge x \neq F$ [Al menos existe un nodo que se encuentra conectado a nodos no especiales]
4. $\forall n, n \in A \Rightarrow ge(n) \geq 1 \wedge gs(n) \geq 1$ [En general, la red se compone por nodos cuyo grado es mayor ó igual a 2]

Las primeras dos características son las que aseguran que habrá actividades ejecutables de forma paralela, mientras que la tercera hace que existan secuencias. De esta manera, es que las redes mixtas contienen tanto relaciones de *precedencia* como de *concurrencia*.

Nótese lo siguiente:

- Si se prescinde de la tercer característica y las primeras dos se restringen solamente a los nodos S y F, se obtienen redes de tipo estrictamente paralelo.
- Si se prescinde de las primeras dos características se obtienen redes secuenciales de al menos tres nodos; si se desea obtener redes secuenciales más pequeñas, serían necesario prescindir de todas las características excepto la última, y ésta tendría que restringir los grados de entrada y salida, de tal forma que sean exactamente 1.
- Si solamente se deja la última restricción, nos quedamos con la definición pura de una red de actividades

Debido a esta observación y al hecho de que se producen relaciones tanto de precedencia como concurrencia, las redes mixtas se pueden ver como una *combinación* o caso general de los primeros dos tipos establecidos.

Con respecto a esta última clase, resulta más complicado encontrar maneras de "reducir" las redes para ver cómo casos pequeños pueden absorber redes más extensas. Y si bien esto es factible, probablemente sea más sencillo justificar la selección de los escenarios de prueba de redes mixtas con base a la clasificación que va de acuerdo al tipo de relaciones.

4.1.2. Clasificación con base al tipo de relaciones

En la sección previa, se mencionó de manera breve el hecho de que cada tipo de red (secuencial, estrictamente paralela, concurrente) cuenta ya sea con 1) solamente relaciones de precedencia, 2) solamente con relaciones de concurrencia ó 3) con ambas. Esto da pie para

que, además de hacer una clasificación con respecto a la topología de los grafos, también se pueda establecer un agrupamiento acorde con el tipo de relaciones presentadas.

De esta manera:

1. Una red secuencial cuenta con sólo relaciones de precedencia. Es decir, cualesquiera dos actividades pertenecientes al grafo se encuentran relacionadas de esta forma:
   - $\forall a, b\ a, b \in A\,[(a\ r\ b) \wedge a \neq b] \Rightarrow r = \prec$
2. Una red estrictamente paralela cuenta sólo con relaciones de concurrencia. Por tanto, se puede visualizar como el complemento de la red secuencial:
   - $\forall a, b\ a, b \in A\,[(a\ r\ b) \wedge a \neq b] \Rightarrow r \neq \prec$
3. Una red concurrente contiene tanto relaciones de precedencia como de concurrencia:
   - $\forall a, b\ a, b \in A\,[(a\ r\ b) \wedge a \neq b] \Rightarrow r = \prec\ \vee\ r \neq \prec$

Probablemente, aquí la cuestión central sería: *¿Qué es lo que esto implica, y por qué es que se han formulado estas clasificaciones?* Esta partición que coloca a cada red en uno de los tres grupos mencionados se ha hecho debido a que—visto desde la perspectiva del razonamiento en general—el tratar con una red secuencial implica trabajar cuando se tienen *hechos* en la base de conocimientos, al igual que lidiar con una red estrictamente paralela conlleva a laborar solamente con las *inferencias* generadas, y—por extensión—trabajar con una red mixta implica tratar con una *combinación* de hechos e inferencias. En ese sentido, lo que nos interesa es *ver que el mecanismo funcione con estos tres casos*. En el primero, como ya vimos, nos basta con probar una red secuencial de dos nodos (ya que por añadidura los demás casos se derivan de éste), al igual que con el segundo (puesto que casos más grandes se pueden descomponer a un grafo con dos nodos concurrentes). Con el tercero, si bien existen un sinnúmero de combinaciones posibles, debido a que lo relevante es ver el comportamiento del algoritmo en un caso donde se le presentan hechos e inferencias, se considera como suficiente hacer las pruebas con una sola y sencilla red también. En este aspecto, se puede considerar que probar con más redes de este tipo hace énfasis en revisar si el *software* creado más que el *concepto* propuesto.

Ahora bien, ¿cómo asegurar que *cualquier* red de actividades caiga en alguna de estas tres categorías? Esto se puede ver desde la propia *definición* del grafo; puesto que a ningún nodo n (salvo, claro está, la fuente y el sumidero) se le permite tener $ge(n) = 0$ y/ó $gs(n) = 0$, entonces en el caso más simple, donde se tiene el mínimo de arcos de entrada y de salida, se obtendrán redes secuenciales. Siguiendo esta misma línea, si se comienzan a variar los grados de entrada y salida de los nodos (aunque no sea de todos), se tendrá que la red es mixta, ya que al menos un nodo poseerá $ge(n) \geq 1$ ó $gs(n) \geq 1$. Cuando sólo se aumenta el grado de S y/o F, se cae en el tipo de red estrictamente paralela. De esta forma, no resulta posible tener redes que caigan fuera de las clases anteriormente establecidas.

Por otra parte, con respecto a los casos específicos de prueba, para cada escenario encuentra definido un conjunto de éstos, los cuales consistieron en asignar una fecha de monitoreo (que estuviera dentro del conjunto previamente establecido), así como estados para las actividades (NS, S, F) y tiempos reales de inicio y/o terminación. De esta forma, resulta posible apreciar varias facetas del mecanismo planteado con una misma red.

Cabe destacar que las pruebas hacen énfasis en aquellos escenarios en donde se utiliza el Razonamiento Temporal.

4.2. Procedimiento seguido para la ejecución de las pruebas

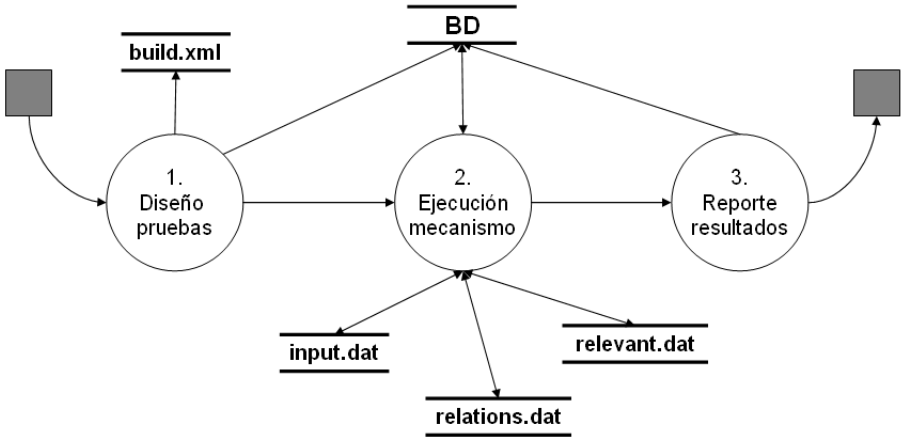

Figura 4.3: Proceso general para la realización de pruebas

El proceso que se siguió para realizar las pruebas (el cual se ilustra en la Figura 4.3, utilizando para ello algunos elementos pertenecientes a los diagramas de flujo de datos [15]) consistió de los pasos que se enumeran a continuación:

1. Diseño de la red de actividades (formulación de los escenarios)
2. Introducción de la red en la base de datos
3. Ajuste de parámetros y valores para ejecutar pruebas sobre el escenario/caso específico
4. Generación de archivo para TimeGraph II
5. Invocación al módulo de Razonamiento Temporal

6. Ejecución del algoritmo de monitoreo
7. Revisión de resultados en consola
8. Revisión de resultados en base de datos
9. Reporte de resultados

Estos pasos se explican enseguida de manera breve. Cabe destacar que este proceso se puede ver en tres fases: 1) diseño de las pruebas, 2) ejecución del mecanismo, 3) observación y registro de resultados.

4.2.1. Diseño

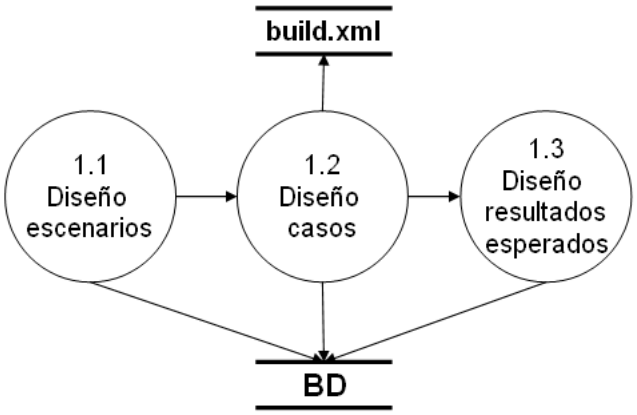

Figura 4.4: Proceso de diseño de pruebas

El diseño de la red consiste simplemente en crear conceptualmente un grafo tipo PERT con actividades relacionadas entre sí, llenar los "slots" más significativos—es decir, que ayudaran a definir—del proyecto y de cada actividad involucrada en éste de acuerdo a los criterios anteriormente explicados, de tal forma que resulte posible observar cómo se comporta el mecanismo. Es así que cada escenario se formula contemplando la siguiente información:

- Ilustración de la red
- Actividades
  - id
  - d (duración)
  - es, ef (tiempos tempranos)
  - ls, lf (tiempos tardíos)

    - cp (pertenencia a la RC)
- Conjunto de milestones (M)
- Conjunto de puntos de monitoreo (PM)
- Relaciones de precedencia
- Encargados de la actividad

Sin embargo, no basta sólo con definir el escenario de prueba. He ahí la necesidad de delinear o *configurar* los casos específicos con el propósito de probar el mecanismo de manera más holística. Para ello, cada escenario cuenta con sus propios casos, y cada uno de éstos a su vez cuenta con su propia diseño. Éste contempla los siguientes parámetros:

- Punto de tiempo actual (t)
- Actividad
    - rs, rf (tiempos reales)
    - estado

De esta forma, resulta posible obtener diversos casos a partir de una misma definición de escenario. Como se verá más adelante, el partir cada escenario en casos resulta útil para evaluar el comportamiento del algoritmo planteado.

Además de diseñar los escenarios y los casos de prueba, es necesario también delinear los resultados esperados, tanto en general como para cada situación particular. Para ello, se establecieron varios *rubros* que serían puntos de revisión, con el objetivo de comparar los *valores* esperados a arrojar por el mecanismo con los valores que finalmente arrojaría. Esto permite de alguna manera ver si el mecanismo propuesto funciona.

Los rubros considerados para dicha sección son los siguientes:

- Acción
- Status de actividades
- Encargados (se supuso que los encargados eran agentes inteligentes)
- Correspondencia con resultado esperado

Cabe destacar que la introducción de los escenarios en la base de datos y los casos se realiza de manera manual, y por cada caso a ser ejecutado se necesita ajustar los parámetros, lo cual también se hace de manera no automática, a través de modificar el archivo `build.xml` de Apache Ant.

4.2.2. Ejecución del mecanismo

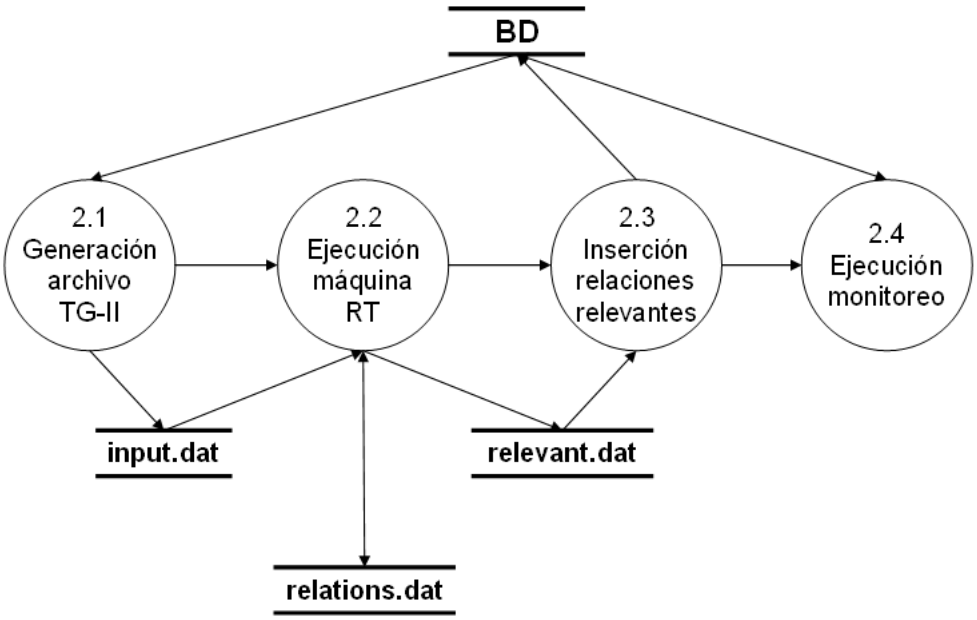

Figura 4.5: Proceso de ejecución del mecanismo

Como se puede apreciar en la Figura 4.5, antes de poder ejecutar la máquina de inferencias de Razonamiento Temporal sobre una red de actividades es necesario crear un archivo que alimente la configuración de ésta al paquete TG-II. Para ello, se extrae el flujo del proyecto desde la base de datos y se transfiere a un formato entendible para Timegraph. Este archivo recibe el nombre de `input.dat`.

Al alimentar la base de conocimientos de TG-II ya resulta posible trabajar con la herramienta para obtener las relaciones relevantes—lo cual a final de cuentas es el dato que interesa. Para ello, se invoca desde un programa de Java un módulo de LISP, el cual a su vez invoca a TG-II, y éste arroja en primera instancia todo el conjunto de relaciones que existen entre los intervalos especificados de entrada. Este conjunto se guarda en un archivo llamado `relations.dat`. Una vez contando con éste, se procede a realizar el filtrado sobre las relaciones y se dejan en el archivo `relevant.dat` solamente las relaciones relevantes. La

información de este archivo posteriormente se pasa a la base de datos. Todo este proceso se realiza mayormente de manera automática.

Posterior al proceso de configuración, se ejecutan propiamente las corridas al algoritmo de monitoreo.

4.2.3. Observación y registro de resultados

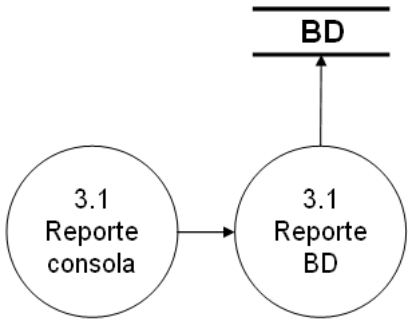

Figura 4.6: Proceso de reporte de resultados

Después de cada corrida, se reportan los resultados en consola. Entre los datos arrojados por este medio, los más significativos indican los pasos que se van llevando a cabo por la mecánica, la detección de discrepancias (en caso de presentarse éstas, adicionalmente se reporta en qué actividades se dieron y cuál es el estatus de las mismas).

Específicamente, los datos arrojados hacia la pantalla son los siguientes:

1. Escenario de prueba
2. Caso de prueba
3. Fecha actual (punto de monitoreo)
4. Constitución de la red de actividades
   - *a*) id actividad
   - *b*) d (duración)
   - *c*) es, ef (tiempos tempranos de inicio y terminación)
   - *d*) ls, lf (tiempos tardíos de inicio y terminación)
   - *e*) fl (holgura)
   - *f*) Conjunto de predecesores de la actividad

5. Milestone más cercano
6. Actividad relacionada al milestone
7. Detección de discrepancias
   - Actividad aislada
     a) Estado actual vs. estado actual
   - Relaciones relevantes vs. relación actual (por cada predecesor)
8. Resultados monitoreo
   - No hay discrepancias
     a) Mensaje de retroalimentación
     b) Semi-intervalos
     c) Vector de relaciones actualizado
   - Existen discrepancias
     a) Mensaje de notificación
     b) Status de actividades (1=atraso, 2=adelanto, 3=normal)
     c) Encargados a quien se realiza la notificación

Asimismo, los resultados de las pruebas se almacenan en la base de datos; por cada prueba, existe un registro que contiene: 1) el caso de prueba el que pertenece, 2) la actividad en la que se detectó una discrepancia, 3) el estatus (N=normal, AT=atrasada, AD=adelantada) de la actividad.

Sin duda, el resultado más importante (el cual determina si se pasó satisfactoriamente la prueba realizada) es la detección de las discrepancias, aunada al hecho de que también se detecte correctamente el tipo de discrepancia presentado en las actividades. Los demás datos, a pesar de que son relevantes, más bien sirven con el propósito de ubicar el caso de prueba y mostrar que el mecanismo funciona (desde el punto de vista del *software*).

4.3. Escenarios de prueba

Teniendo en cuenta todo lo anterior, enseguida se describen los escenarios de prueba junto con sus respectivos casos.

Para cada escenario, se proporciona un nombre que a grandes rasgos lo define y asimismo se proporciona una descripción de la red (ilustración, características de las actividades, relaciones de precedencia). Aunado a esto, se enumeran los casos de prueba derivados a partir de dicho escenario; para cada uno de éstos, se establecen dos parámetros, que son el *tiempo actual* y el *estado* de las actividades (si ya iniciaron, están en proceso o si ya se terminaron, y en qué fechas ocurrió esto).

4.3.1. Escenario 1: Red secuencial de una sola actividad

Si bien este escenario es de una naturaleza muy simple por componerse de sólo una actividad y debido a que las relaciones de precedencia se sostienen exclusivamente con los nodos de inicio y terminación, al mismo tiempo se puede decir que sí es importante tomarlo en cuenta, puesto que se puede considerar como un caso de *base*, con el cual es posible obtener una especie de "cota inferior" para el algoritmo (es decir, si no puede ni siquiera resolver de manera correcta este caso, su desempeño se pondría en tela de juicio). Además de esto, su relevancia consiste en que no sería poco común encontrarse ante un escenario con una actividad *global*, ya que existen diferentes niveles de detalle dentro de las redes de actividades [19].

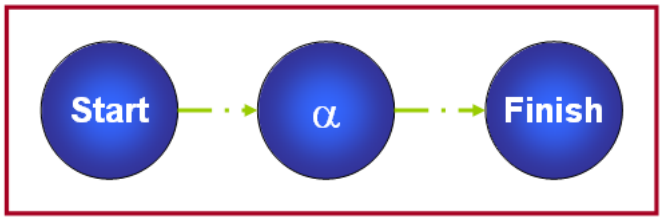

Figura 4.7: Escenario 1

Actividades

Actividad (duración)	es	ef	ls	lf	cp
α (20)	0	20	0	20	T

$M = \{5, 15\}, PM = \{0, 12, 14\}$

Relaciones

Para este escenario no existen relaciones.

Agentes

Agente	Actividades asignadas
`agent_alpha`	α
`agent_project`	α

Casos de prueba

1. α se atrasa
2. α se adelanta
3. α a tiempo

Parámetros

Caso 1

$t = 0$

Actividad	**Estado**	**rs**	**rf**
α	NS	N/A*	N/A*

* Cuando el tiempo real de inicio o de terminación no cuente con valor alguno, se utilizarán las siglas N/A.

Caso 2

$t = 12$

Actividad	**Estado**	**rs**	**rf**
α	F	0	10

Caso 3

$t = 14$

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A

Resultados

Caso	**Descripción**	**Acción**	**Status**	**Agentes**	**Esperado**
c1	α se atrasa	N/A	$\alpha = \text{AT}$	`agent_alpha` `agent_project`	Sí
c2	α se adelanta	N/A	$\alpha = \text{AD}$	`agent_alpha` `agent_project`	Sí
c3	α a tiempo	N/A	$\alpha = \text{N}$	Ninguno	Sí

4.3.2. Escenario 2: Red secuencial de dos actividades

Una cuestión que vale la pena recalcar es por qué este caso (y de manera análoga el siguiente) no es equivalente al Caso 1, siendo que se trata de detectar a una actividad que se atrasa. La principal razón para hacer la distinción entre ambas pruebas es que—como se verá más adelante—no resulta lo mismo detectar discrepancias en actividades que pertenecen

a la RC que en actividades no pertenecientes a ella (simplemente, recordemos que éstas últimas cuentan por definición con una holgura).

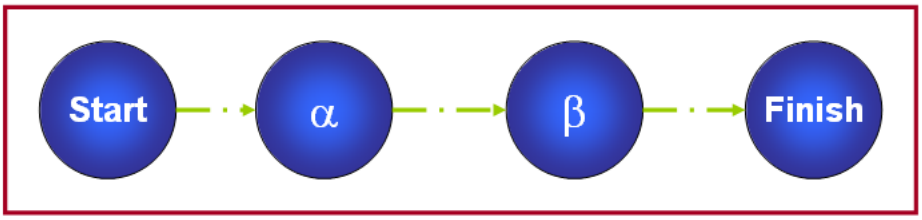

Figura 4.8: Escenario 2

Actividades

Actividad (duración)	es	ef	ls	lf	cp
α (6)	0	6	0	6	T
β (7)	6	13	6	13	T

$M = \{10\}, PM = \{3, 5, 9, 12\}$

Relaciones

α (m) β

Agentes

Agente	Actividades asignadas
`agent_alpha`	α
`agent_beta`	β
`agent_project`	α, β

Casos de prueba

1. α se atrasa
2. α se adelanta
3. β se atrasa
4. β se adelanta

Parámetros

Caso 1

$t = 3$

Actividad	Estado	rs	rf
α	NS	N/A	N/A
β	NS	N/A	N/A

Caso 2

$t = 5$

Actividad	Estado	rs	rf
α	F	0	4
β	NS	N/A	N/A

Caso 3

$t = 9$

Actividad	Estado	rs	rf
α	F	0	6
β	NS	N/A	N/A

Caso 4

$t = 12$

Actividad	Estado	rs	rf
α	F	0	6
β	F	6	10

Resultados

Caso	Descripción	Acción	Status	Agentes	Esperado
c1	α se atrasa	N/A	$\alpha = \text{AT}$ $\beta = \text{N}$	`agent_alpha` `agent_project`	Sí
c2	α se adelanta	N/A	$\alpha = \text{AD}$ $\beta = \text{N}$	`agent_alpha` `agent_project`	Sí
c3	β se atrasa	N/A	$\alpha = \text{N}$ $\beta = \text{AT}$	`agent_beta` `agent_project`	Sí
c4	β se adelanta	N/A	$\alpha = \text{N}$ $\beta = \text{AD}$	`agent_beta` `agent_project`	Sí

4.3.3. Escenario 3: Red estrictamente paralela

El escenario 3 puede considerarse como el más sustancial de los tomados en cuenta para realizar los experimentos, puesto que es en el que se aplica (y se aplica básicamente de manera exclusiva) el Razonamiento Temporal para diagnosticar los atrasos y adelantos. Por lo tanto, los casos de prueba son definitivos para proveer una idea general que permita decir a grandes rasgos si funciona o no y por qué. Debido a esta misma razón, fue en este escenario en el que se hicieron más pruebas, procurando que éstas fueran lo suficiente comprensivas como para

brindar una panorámica del mecanismo en acción y al mismo tiempo, como ya se mencionó, sirvieran para en parte determinar si puede cumplir con el diagnóstico de discrepancias.

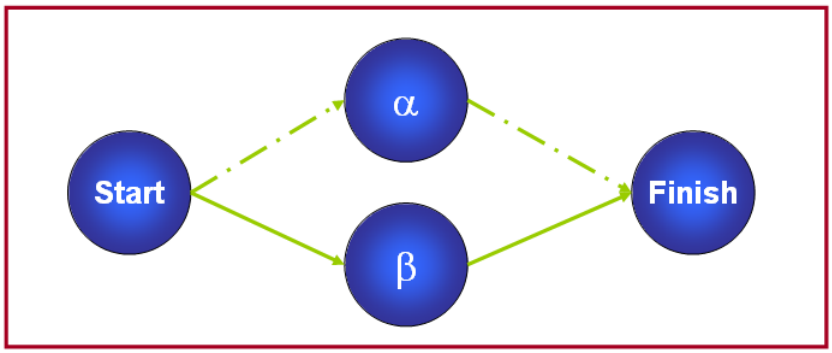

Figura 4.9: Escenario 3

Actividades

Actividad (duración)	es	ef	ls	lf	cp
A(10)	0	10	0	10	T
B(4)	0	4	6	10	F

$M = \{11\}, PM = \{0, 1, 3, 7, 8, 9\}$

Relaciones

Agentes

Agente	Actividades asignadas
`agent_alpha`	α
`agent_beta`	β
`agent_project`	α, β

Casos de prueba

1. α se atrasa
2. α se adelanta
3. β se atrasa
4. β se adelanta
5. α se adelanta y β se atrasa
6. α se atrasa y β se adelanta

7. α se atrasa y β se atrasa

8. α se adelanta y β se adelanta

9. α a tiempo y β a tiempo (no existen discrepancias)

Parámetros

Caso 1

$t = 0$

Actividad	**Estado**	**rs**	**rf**
α	NS	N/A	N/A
β	NS	N/A	N/A

Caso 2

$t = 8$

Actividad	**Estado**	**rs**	**rf**
α	F	0	7
β	S	5	N/A

Caso 3

$t = 9$

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A
β	S	7	N/A

Caso 4

$t = 10$

Actividad	**Estado**	**rs**	**rf**
α	F	0	10
β	F	0	3

Variación 1

$t = 7$

Actividad	**Estado**	**rs**	**rf**
α	F	0	10
β	F	0	3

Caso 5

$t = 8$

Actividad	**Estado**	**rs**	**rf**
α	F	0	5
β	S	7	N/A

Caso 6

$t = 3$

Actividad	**Estado**	**rs**	**rf**
α	S	2	N/A
β	F	0	2

Caso 7

$t = 3$

Actividad	**Estado**	**rs**	**rf**
α	S	5	N/A
β	S	3	N/A

Caso 8

$t = 9$

Actividad	**Estado**	**rs**	**rf**
α	F	0	7
β	F	6	8

Caso 9

El presente caso—que fue ejecutado de manera secuencial—se encuentra dividido en tres partes, donde la diferencia entre cada una radica en el estado de β y el punto de monitoreo utilizado (el estado de α no varía debido a que esto generaría discrepancias, y lo que se está intentando mostrar es el comportamiento donde éstas no se encuentran presentes). Se diseñó de esta manera con el propósito de observar cómo, a través del tiempo (es decir, pasando por los puntos de monitoreo establecidos), el mecanismo se encarga de ir verificando las relaciones relevantes entre las actividades y las actualiza. Asimismo, el caso—al igual que en los escenarios anteriores—se ha puesto para ver si el mecanismo funciona cuando todo es "normal".

Parte 1

$t = 1$

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A
β	NS	N/A	N/A

Parte 2

$t = 3$

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A
β	S	3	N/A

Parte 3

$t = 8$

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A
β	F	3	7

Resultados

Caso	Descripción	Acción	Status	Agentes	Esperado
c1	α se atrasa	1	α = AT β = N	`agent_alpha` `agent_project`	Sí
c2	α se adelanta	6	α = AD β = N	`agent_alpha` `agent_project`	Sí
c3	β se atrasa	4	α = N β = AT	`agent_beta` `agent_project`	Sí
c4	β se adelanta	9	α = N β = AD	`agent_beta` `agent_project`	Sí
c4.v1	β se adelanta	5	β = N α = N	Ninguno	No

c5	α se adelanta,
	β se atrasa
		6	α = AD
			β = AT
				`agent_alpha`
				`agent_beta`
				`agent_project`
					Sí

c6	α se atrasa,
	β se adelanta
		5	α = AT
			β = AD
				`agent_alpha`
				`agent_beta`
				`agent_project`
					Sí

c7	α se atrasa,
	β se atrasa
		4	α = AT
			β = AT
				`agent_alpha`
				`agent_beta`
				`agent_project`
					Sí

c8	α se adelanta,
	β se atrasa
		9	α = AD
			β = AT
				`agent_alpha`
				`agent_beta`
				`agent_project`
					Sí

c9.p1	α a tiempo, β a tiempo	2	α = N β = N	Ninguno	Sí
c9.p2	α a tiempo, β a tiempo	4	α = N β = N	Ninguno	Sí
c9.p3	α a tiempo, β a tiempo	5	α = N β = N	Ninguno	Sí

4.3.4. Escenario 4: Relaciones de precedencia y concurrencia

El último escenario—que, como ya se vio, cubre los casos en los que hay tanto relaciones de precedencia como de concurrencia—se encuentra planteado con el fin de mostrar el funcionamiento del mecanismo cuando debe lidiar tanto con *hechos* como con *inferencias*.

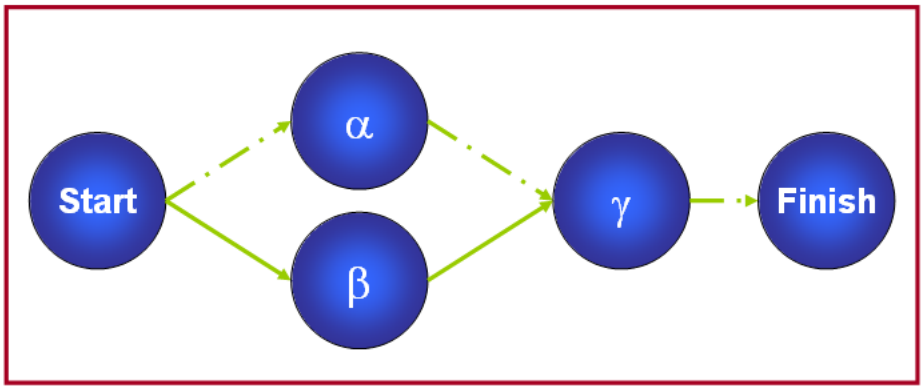

Figura 4.10: Escenario 4

Actividades

Actividad (duración)	es	ef	ls	lf	cp
α (13)	0	13	0	13	T
β (11)	0	11	2	13	F
γ (5)	13	18	13	18	T

$M = \{18\}, PM = \{8, 11, 15\}$

Relaciones

α (m) γ
β (b m) γ

Agentes

Agente	Actividades asignadas
`agent_alpha`	α
`agent_beta`	β
`agent_gamma`	γ
`agent_flow1`	α, γ
`agent_flow2`	β
`agent_project`	α, β, γ

Casos de prueba

1. γ se atrasa
2. α se adelanta
3. β se adelanta
4. γ se adelanta

5. Todas las actividades a tiempo (no hay discrepancias)

Parámetros

Caso 1

$t = 15$

Actividad	**Estado**	**rs**	**rf**
α	F	0	13
β	F	2	13
γ	NS	N/A	N/A

Variación 1

Actividad	**Estado**	**rs**	**rf**
α	F	0	13
β	S	2	N/A
γ	NS	N/A	N/A

Variación 2

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A
β	F	2	13
γ	NS	N/A	N/A

Caso 2

$t = 11$

Actividad	**Estado**	**rs**	**rf**
α	F	0	10
β	S	1	N/A
γ	NS	N/A	N/A

Caso 3

$t = 8$

Actividad	**Estado**	**rs**	**rf**
α	S	0	N/A
β	F	0	7
γ	NS	N/A	N/A

Caso 4

$t = 17$

Actividad	**Estado**	**rs**	**rf**
α	F	0	13
β	F	0	11
γ	F	13	16

Caso 5

$t = 8$

Actividad	Estado	rs	rf
α	S	0	N/A
β	S	0	N/A
γ	NS	N/A	N/A

Resultados

Caso	Descripción	Acción	Status	Agentes	Esperado

c1	γ se atrasa	N/A	α = N
			β = N
			γ = AT
				`agent_gamma`
				`agent_flow1`
				`agent_project`
					Sí

c1.v1	γ se atrasa (por β)	5	α = N
			β = AT
			γ = AT
				`agent_gamma`
				`agent_beta`
				`agent_flow1`
				`agent_project`
					Sí

c1.v2	γ se atrasa (por α)	6	α = AT
			β = N
			γ = AT
				`agent_gamma`
				`agent_alpha`
				`agent_flow1`
				`agent_project`
					Sí

c2	α se adelanta	5	α = AD
			β = N
			γ = N
				`agent_alpha`
				`agent_flow1`
				`agent_project`
					Sí

c3	β se adelanta	6	α = N β = N γ = N	Ninguno	No

c4	γ se adelanta	9	α = N
			β = N
			γ = AD
				`agent_gamma`
				`agent_flow1`
				`agent_project`
					Sí

c5	α a tiempo, β a tiempo, γ a tiempo	4	α = N β = N γ = N	Ninguno	Sí

4.4. Discusión de Resultados

Tomando en cuenta el conjunto de pruebas presentado, el mecanismo logró arrojar los resultados esperados en la mayoría de los casos: 1) se detectó la presencia/ausencia de discrepancias de acuerdo al caso planteado (salvo dos casos que a se comentan posteriormente), 2) se detectó correctamente el tipo de discrepancia presentada (atraso o adelanto) y en qué actividad fue que se dio, 3) resultados menos relevantes (punto de revisión correspondiente,

actividad relacionada con el punto de revisión, obtención del conjunto de encargados a notificar, acción correspondiente, actualización en las relaciones) también se obtuvieron conforme a lo esperado.

En cuanto a los dos casos en los que el mecanismo no se comportó conforme a lo esperado, es posible enmarcarlos dentro de una misma situación. Ésta se puede visualizar como una forma de "engaño" para el algoritmo, puesto que éste detecta una relación que se encuentra dentro del conjunto permitido, pero no es capaz de ver que por las duraciones se genera una discrepancia. En otras palabras, el engaño consiste en presentar al algoritmo una relación actual que es *permitida* pero que en realidad contiene una discrepancia temporal. Sin embargo, este caso se da específicamente cuando existe un *adelanto* en una actividad que *no* pertenece a la ruta crítica. En la sección de Trabajos Futuros se vuelve a comentar este detalle.

Los demás casos, por otro lado, debido a que presentan la discrepancia al crearse una relación extraña al conjunto teórico, hacen posible que se identifique el atraso o adelanto. Por tanto, se puede decir que en ese aspecto el mecanismo es *completo* en cuanto a la detección de atrasos se refiere.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

Con base al desarrollo de la presente investigación, se ha podido llegar a las siguientes conclusiones:

- El RT *sí* puede utilizarse para la detección de discrepancias de origen temporal que se presentan en las actividades de un proyecto. Como se vio en los resultados la Sección 4.3, es posible que mediante aplicar los conceptos derivados de dicho razonamiento se puedan identificar aquellas situaciones—con respecto a la dimensión temporal—que se encuentran fuera de lo planeado.

- Sin embargo, el RT compone *sólo una parte* del mecanismo construido para realizar el seguimiento: el uso de este tipo de razonamiento *per sé* no es suficiente para proveer una infraestructura de monitoreo en las actividades del proyecto. Es decir, aunado a las relaciones de precedencia entre las actividades (las cuales se toman como entrada para la máquina de inferencias de RT), existen otros datos que resultan *necesarios* para identificar las discrepancias temporales que pueden presentarse a lo largo del desarrollo del proyecto (por ejemplo, la duración, así como los tiempos de inicio y terminación son importantes para que el mecanismo de monitoreo funcione correctamente). Finalmente, se puede decir que estos datos complementan el trabajo hecho por la máquina de inferencias de RT, y todo en conjunto constituye el mecanismo presentado.

- A pesar de los costos adicionales que pudiera representar para el proceso de monitoreo, el RT brinda una perspectiva diferente de las actividades, ya que las considera no como entidades individuales, sino como *entidades entrelazadas*, tanto por relaciones de precedencia (lo cual es un dato proporcionado por la constitución de la red) como por relaciones de concurrencia. Éstas últimas contribuyen a detectar actividades paralelas, las cuales sirven para propósito de notificaciones; de esta manera, es posible determinar qué actividades son concurrentes a aquellas en las cuales se ha detectado un atraso o adelanto, y así notificarles esta situación a todos los involucrados correspondientes para realizar los ajustes requeriedos. Teniendo esto en cuenta y yendo un poco más lejos, en cierta medida el uso de RT podría contribuir parcialmente a llevar el proceso de control del proyecto de manera automática, mediante el uso de agentes inteligentes [25] que se

comuniquen entre sí y lleven a cabo mecanismos necesarios para realizar el control (por ejemplo, negociación). Parte de esto se discute en la sección de Trabajos Futuros.

- Adicionalmente, a lo largo de la presente investigación, se pudo responder a las interrogantes planteadas en un principio:
  - La representación temporal que cubrió las necesidades planteadas en el objetivo general (detectar discrepancias de origen temporal en un proyecto) consistió en una combinación de intervalos y puntos de tiempo. Sin embargo, la entidad primeramente mencionada resultó vital para el desarrollo del mecanismo; por otra parte, conviene recalcar—como ya se había mencionado previamente—que el razonamiento solamente se hizo sobre los intervalos de tiempo (si bien se utilizaron puntos con fines de *representación*, con éstos no se realizó ningún tipo de operación algebraica).
  - Asimismo, la herramienta que fue capaz de manejar el razonamiento con respecto a la representación temporal seleccionada fue el sistema de razonamiento temporal TimeGraph II, puesto que soporta RT basado en intervalos, tiene un buen comportamiento en cuanto a complejidad temporal y espacial (esto con respecto a los casos probados a lo largo de la investigación), se encuentra disponible, es sencillo de usar, se acopla con la plataforma de Allegro Common Lisp y es una de las herramientas más recientes dentro del área.
  - Por otra parte, como ya se vio en el capítulo 3, fue posible el diseñar e implementar un algoritmo que pudiera llevar a cabo la tarea de monitoreo de actividades con respecto al tiempo y que de igual manera pudiera trabajar en conjunto con el enfoque y herramienta seleccionados.
  - Por último, también resultó posible aplicar un conjunto de casos de prueba al mecanismo construido y observar su comportamiento. Con base en los resultados de estas pruebas, se concluyó que el mecanismo sí puede detectar los atrasos, adelantos y situaciones normales presentados, mas no en todos los casos, puesto que existen maneras de "engañar" al algoritmo actual.

5.2. Trabajo Futuro

Algunas de las áreas en las que se pueden desarrollar trabajos a futuro sobre la presente investigación son las siguientes:

- Mejoras/extensiones al mecanismo *per sé*
- Incorporación del proceso de control
- Integración a otros sistemas

5.2.1. Mejoras y extensiones al mecanismo de monitoreo

Sin duda, uno de los puntos de mejora incluye el encontrar una manera de conseguir que el mecanismo detecte aquellos casos en los que se ha visto que no identifica las discrepancias

temporales presentadas. Esto podría lograrse a través de realizar validaciones adicionales o la inclusión de más intervalos a la red de actividades. No obstante, el trabajo futuro precisamente consistiría en buscar cómo resolver este problema.

Una mejora interesante consta de incluir los porcentajes de avance en las actividades. De esta manera, sería factible hacer una proyección más precisa en cuanto a la duración "actual" de los intervalos y producir relaciones actuales que reflejen mejor el estado del proyecto.

Del mismo modo, una extensión sustancial consistiría en relajar las restricciones impuestas a la red de actividades, lo cual implicaría:

- Trabajar con dependencias de otros tipos (suaves, de inicio a inicio, fin a fin, etc.)
- Considerar la representación AOA
- Trabajar con redes que contengan actividades de retroalimentación
- Probar redes que contengan distintos niveles de "jerarquía" (por ejemplo, el administrador del proyecto ve las actividades más generales, mientras que el jefe de un equipo en específico ve todas las tareas que componen dichas actividades).

Por otra parte, actualmente aquellas actividades "aisladas"—las que no cuentan con relaciones relevantes—- se monitorean mediante la verificación de estados (recordemos el uso de la función CHECK-STATE). Se ha visto que algunos de estos casos (sobretodo aquellos que pertenecen a las redes secuenciales) podrían cubrirse también con RT, en vez de utilizar esta función. Es decir, podrían incluirse dentro del conjunto de relaciones relevantes las relaciones de precedencia remota y determinar si una actividad dada cumple con éstas al considerar al sus predecesores como un solo intervalo. Por tanto, la extensión consistiría en cambiar la implementación para buscar que aún los casos donde no hay relaciones concurrentes se cubran también con RT (aunado a esto, sería posible asimismo evaluar si esta forma de abordar este tipo de casos es mejor que la verificación por estados).

De igual manera, como ya se había explicado anteriormente, la actualización de relaciones se realiza mediante el uso de semi-intervalos. No obstante, el inconveniente de éstos consiste en que también admiten relaciones que—por las características de las actividades—se sabe que probablemente no se darán. En consecuencia, se podría intentar hacer la actualización mediante dejar solamente la relación que, de acuerdo con los datos proporcionados por las actividades, sería la que se daría. De esta manera, se tendría un conjunto más reducido de relaciones posibles y tal vez la detección de discrepancias sería más precisa.

Otra extensión sería el representar a la red de actividades ya no con una red de intervalos, sino con un modelo cuantitativo-cualitativo, el cual toma tanto la información métrica (puntos de tiempo, posicionamiento numérico de los eventos en el tiempo, etc.) como la información

con referencia a los vectores de relaciones existentes entre los nodos del grafo, además de que conjunta en un mismo grafo puntos e intervalos de tiempo. Si se recuerda, la red de intervalos definida por Allen solamente considera las *relaciones* cualitativas entre los nodos y no representa puntos; por tanto, tal vez la inclusión de este tipo de restricciones en la red podría proveer un enfoque más holístico y provechoso para el proceso de monitoreo. Un ejemplo de este tipo de red se puede ver en Figura 5.1. (Poner referencia al paper...)

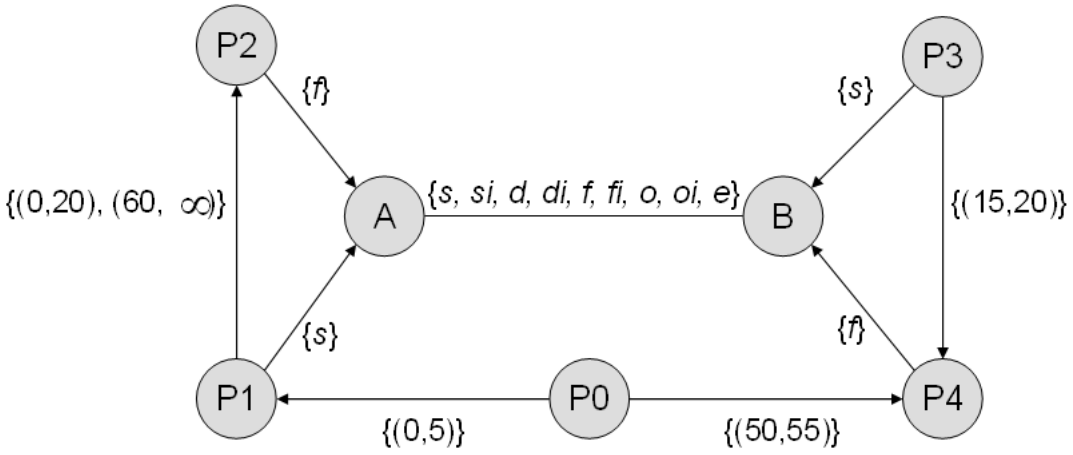

Figura 5.1: Red de restricciones cualitativas y cuantitativas. A y B son intervalos, mientras que P0, P1, P2, P3, P4 y P5 son puntos.

5.2.2. Incorporación de la tarea de control

De acuerdo a [23], existen cinco grupos de procesos que intervienen en la administración de proyectos:

- Apertura
- Planeación
- Ejecución
- Seguimiento y control
- Clausura

Como se puede ver, el proceso de seguimiento (monitoreo) se encuentra íntimamente ligado al control en un proyecto. Es por ello que el presente trabajo, al abordar con mayor énfasis la tarea de monitoreo, deja un espacio abierto para posteriormente tratar con la tarea de control, que toma relevancia por la estrecha relación que existe entre ambos procesos. En

ese aspecto, el proceso de seguimiento puede considerarse como "detectar las discrepancias" y el proceso de control como "tomar acciones al respecto".

Ahora bien, surge la interrogante: *¿cómo puede efectuarse el control a partir del mecanismo propuesto?*. Hasta ahora, éste detecta las discrepancias temporales, genera una notificación con la situación identificada y obtiene *quiénes* pudiesen o les convendría estar interesados en recibir dicha notificación (recordando también que los interesados pueden ser directamente personas o agentes que las representan). A partir de aquí, algunas de las alternativas viables son las siguientes: 1) construir un mecanismo que solamente *distribuya* las notificaciones (o integrar el mecanismo de monitoreo a uno ya existente, como ya se discutió previamente) y dejar que el control de cambios se realice de forma manual, y/o 2) aprovechando que mediante el RT es posible obtener no sólo a los directamente involucrados en una discrepancia temporal sino también a aquellos encargados de las actividades concurrentes, se podría buscar que sean *agentes inteligentes* los que realicen el control de cambios de manera *automatizada*. No obstante, el rango de posibilidades de integración al proceso de control obviamente puede ser más amplio e incluir otras alternativas además de éstas dos.

5.2.3. Integración a otros sistemas

Es lógico suponer que el trabajo desarrollado mediante la presente investigación no es para tratarse como una pieza aislada, sino más bien como un componente utilizable en sistemas de mayor escala, el cual pudiera ser aprovechado para agregar nuevas funcionalidades a lo ya establecido. Como ya se mencionó, uno de estos sistemas podría ser *JITIK*; si bien en el capítulo pasado se discutió el esquema bajo el cual el mecanismo puede ser integrado a la plataforma, el implementar esta integración es un posible trabajo futuro.

5.3. Trabajos relacionados

En la presente sección, se discuten brevemente algunos trabajos que se encuentran ligados en ciertos aspectos a lo realizado por la investigación. Éstos son:

- Monitoreo de actividades de las personas en casas inteligentes [3]
- Formas convencionales de llevar a cabo el proceso de seguimiento [21]
- Control de calendario con equipos pequeños [24]

5.3.1. Monitoreo de las actividades de las personas en casas inteligentes

En regiones donde la población de adultos mayores cada vez va en aumento y es asimismo cada vez más común que éstos vivan solos en sus propios hogares, surge la necesidad de estar al pendiente de sus actividades. De esta manera, los familiares y médicos que los atienden—sobretodo aquellos quienes se encuentran lejos de la casa—pueden asegurarse que se encuentran bien de salud y que no desarrollan actividades peligrosas, tales como dejar una

estufa encendida por equivocación. Para cubrir esta necesidad, se ha desarrollado el concepto de *casas inteligentes*, las cuales tienen como propósito permitir la *autonomía* de las personas mayores y de igual forma preservar su *calidad de vida*.

Dentro del contexto presentado, existen varios puntos importantes a considerar. Por ejemplo, el hecho de que estas casas—mediante sensores—obtengan la información más relevante sobre el estado y actividades de la persona. Otro aspecto es cómo concentrar esta información para *presentarla* de una manera entendible y concisa ante los interesados. Aquí, de manera implícita, se encuentra implicado el manejo del tiempo y razonamiento sobre éste. Para ello—en cuanto al aspecto del tiempo se refiere—, el objeto de principal interés resulta la *ocurrencia* de *eventos* que se dan bajo un cierto *contexto* donde algunas *condiciones* son verdaderas. Al suceder este tipo de eventos bajo las condiciones indicadas, se lleva a cabo una *acción* pertinente. Para manejar estas tres variables (eventos, acciones y condiciones), se utilizan reglas ECA (*Event-Condition-Action*, por sus siglas en inglés). A continuación se muestran algunos ejemplos de este tipo de reglas.

```
on ‘‘persona ha estado en cama por un largo periodo de tiempo’’
   if ‘‘no se espera que esté en cama durante ese periodo’’
                  do ‘‘contactar encargado’’

              on ‘‘presión sanguínea mayor a 200/175
          por más de dos muestras sucesivas en el mismo día’’
if ‘‘régimen médico para controlar la presión alterado recientemente’’
                  do ‘‘notificar a equipo médico’’
```

Por lo tanto, para este proyecto en particular, en cuanto al tiempo se refiere, resulta importante trabajar con eventos instantáneos (puntos de tiempo), así como con referencias temporales que manejen duraciones. De igual manera, otros conceptos temporales importantes considerados incluyen:

- Repetición de eventos
- Composición secuencial
- Eventos progresivos y completados
- Frecuencia de ocurrencias

Al tener todo lo anterior en cuenta, es posible construir mecanismos que se integren a las casas de tal manera que: 1) se detecten los eventos más sobresalientes, 2) estos eventos se puedan interpretar dentro de un contexto (para lo cual se utilizan conceptos de RT) y 3) la información pertinente sea presentada de forma concreta a las personas indicadas.

El proyecto de casas inteligentes plantea una propuesta interesante, y que se puede considerar como análoga en muchos sentidos al mecanismo desarrollado en la investigación, ya que en ambos casos se trata de realizar un *monitoreo* y *avisar* cuando se detecten situaciones

fuera de lo normal. Asimismo, en los dos trabajos se promueve el uso de Razonamiento Temporal.

Sin embargo, las dos propuestas cuentan con elementos que los diferencian ampliamente. En primera instancia, el tipo de RT que se utiliza es diferente; el proyecto planteado por [3] hace uso de la *semántica* temporal, la cual implica—entre otras cosas—el encontrar relaciones entre puntos de tiempo dentro de un *contexto* [26]. Por ejemplo, inferir que si en dos mediciones realizadas el mismo día una persona resulta con presión baja, eso implica que la persona tuvo la presión baja durante ese día (a esto se le conoce como "concatenación" de eventos, o *inferencia horizontal*). Eso en principio es diferente a lo definido por el álgebra de intervalos, aunque en el fondo todo esto sigue siendo inferencia. Por otra parte, en el proyecto de casas inteligentes, los resultados obtenidos son para conformar las reglas que finalmente se utilizan para detectar situaciones anómalas (que no necesariamente son de origen temporal), mientras que el mecanismo de monitoreo utiliza los resultados de la inferencia como datos primarios para identificar las discrepancias temporales. Por tanto, la diferencia principal entre ambos trabajos es el *tipo* de RT usado y asimismo la *utilización* que se les da a los resultados de la inferencia.

5.3.2. Formas convencionales de llevar a cabo el proceso de seguimiento

Algunas maneras que se utilizan comúnmente para darle seguimiento a un proyecto con el fin de *observar* su desempeño (tiempo, costo, alcance, recursos) son las siguientes:

Supervisión.- Observación directa, reportes de avances.

Puntos de revisión.- Representan metas a corto plazo que deben ser cumplidas.

Pruebas y demostraciones.- Mediante evaluar el producto o servicio de manera parcial es posible determinar su avance.

Expertos externos.- Es posible que los consultores, con base a su experiencia, evalúen el progreso en el proyecto.

Estatus de la documentación de diseño.- Entre más documentación se tenga, se infiere que el proyecto va más avanzado.

Utilización de recursos.- De acuerdo al consumo de recursos se puede analizar el desempeño.

Benchmarking (analogía).- Las comparaciones sirven para establecer puntos de referencia.

Cambios, re-trabajo, fallas.- El trabajo adicional definitivamente repercute en el desempeño del proyecto.

Éstas se pueden considerar como tareas generales ("reglas de dedo") que se pueden realizar para estar monitoreando el progreso en un sentido holístico. Sin embargo, para *medirlo*

con respecto a alcance, calidad, tiempo y costos, se cuenta con algunos índices que resultan útiles. Aquellos relacionados con la calendarización (tiempo) se muestran a continuación:

BCWS.- *Costo presupuestado del trabajo calendarizado.* Indica el costo del trabajo que está programado para realizarse.

BCWP.- *Costo presupuestado del trabajo realizado.* Indica el costo del trabajo que ya se ha realizado (en algunos casos, se tiene que calcular de manera parcial).

Con la ayuda de los índices mencionados, resulta posible calcular algunos indicadores que dan la pauta de si el proyecto va a tiempo o no y qué tanto hace falta para que se vaya de acuerdo a lo planeado. Estos indicadores son:

SV.- *Varianza en calendario.* Indica la variación (en términos monetarios) que existe entre lo que, de acuerdo al desarrollo del proyecto, se debería haber gastado y lo que se ha gastado hasta la fecha. Supone que si el balance es negativo, esto implica que existe un retraso en el proyecto. Si es positivo, entonces supone que hay un adelanto en el calendario, y si es igual a cero, que se va a tiempo.

TV.- *Varianza en tiempo.* Indica la cantidad de tiempo que se necesita para que el desarrollo actual alcance al desarrollo planeado (por ejemplo, una semana).

Estos indicadores se calculan con las siguientes fórmulas:

$SV = BCWP - BCWS$
$TV = SD - BCSP$, donde BCSP es el tiempo en el que BCWS=BCWP

Asimismo, es importante mencionar que estos indicadores se revisan junto con otros para tener una visión más integral del progreso en el proyecto.

Ahora bien, para obtener información con mayor detalle, también se pueden analizar las actividades individuales del proyecto, considerando para ello el *índice de desempeño de calendario* (SPI). Si este índice es mayor a uno, se considera que hay un adelanto en la actividad; si se encuentra entre uno y cero, implica que se lleva un retraso; y, si es exactamente igual a uno, es que la actividad va a tiempo.

$$SPI = \frac{BCWP}{BCWS}$$

Sin duda, los métodos convencionales—aún y con las desventajas que pudiesen poseer—son los que tradicionalmente se utilizan debido a los resultados que han producido. En cuanto a esto, el mecanismo de monitoreo propuesto plantea algo nuevo y en cierta medida diferente; además, necesitaría probarse en *ambientes reales* para evaluar su desempeño. No obstante,

el mecanismo no se ha planteado como *sustitución* a estos métodos, y tampoco se considera como mutuamente excluyente con respecto a ellos. De hecho, como se mencionó en secciones anteriores, el incluir porcentajes de avance podría volverlo más proactivo y cercano a las situaciones actuales presentadas dentro del proyecto. Por tanto, el mecanismo puede considerarse como *complementario* a los métodos convencionales.

Asimismo, una diferencia importante entre el mecanismo y estos métodos es que éstos se encuentran fuertemente enfocados hacia la dimensión de *costos*. De hecho, el tiempo se considera en términos del presupesto; el mecanismo, por otra parte, se basa puramente en el concepto del tiempo. De igual manera, el mecanismo se basa en *razonamiento* para detectar atrasos y adelantos, mientras que los métodos convencionales hacen uso de matemáticas, estadística y heurísticas para realizar el seguimiento.

5.3.3. Control de calendario con equipos pequeños

En proyectos de gran escala o donde el tiempo de desarrollo es considerable, el dar seguimiento es un proceso difícil, puesto que el control se puede perder en los niveles más bajos de la estructura de trabajo. Es por ello que en [24], se define un método basado en el desarrollo y supervisión mediante *equipos pequeños* que puedan ir monitoreando el progreso de las tareas que corresponden a distintos módulos del proyecto. Este método se usó para el desarrollo de un sistema de software, proyecto que duró por varios años.

Como punto de partida, este método sugiere la integración de un *equipo guía* que se encargue de apoyar al líder para administrar el proyecto. Por otra parte, los equipos propiamente de trabajo, están conformados por aproximadamente tres miembros cada uno, de los cuales uno de ellos es quien se dedica a realizar las operaciones relacionadas con el diseño y los demás de la implementación de dicho diseño. Todas las decisiones se toman por consenso, y se llevan a cabo *revisiones periódicas* para determinar el estado de las actividades; estas juntas se dan en varios niveles (actividades, módulos, conjunto de módulos). De igual manera, las revisiones constan de tres etapas: aprobación, revisión por compañeros de equipo y revisión final. Si el equipo considera que alguna tarea no se terminará en el tiempo debido, negocía con el administrador del proyecto o miembro del equipo guía y se llega a un acuerdo para estipular una nueva fecha de entrega. Todo esto repercute obviamente en cambios al calendario previamente planeado.

El método planteado en un principio involucra un alto grado de interacción entre todos los miembros de los equipos, y esto trae un costo adicional por la cantidad de juntas y revisiones a realizar durante el proyecto. No obstante, el dar un seguimiento que se encuentra enfocado a los paquetes de trabajo (tareas) más pequeños de manera frecuente y cercana finalmente provoca que la detección de problemas se dé en un tiempo razonable. De igual manera, evita que éstos escalen a niveles superiores por no haber recibido la atención adecuada. Cabe destacar que este método no es formal y fue probado solamente para el desarrollo de un proyecto, en el cual se obtuvieron resultados satisfactorios.

Ahora bien, poniendo este trabajo en contexto con la investigación realizada, se puede decir que el primero—aún y cuando fue desarrollado para un proyecto en el área de sistemas de informática—se encuentra muy enfocado hacia procesos llevados a cabo por *personas*. Es decir, las tareas concernientes al monitoreo del proyecto, seguimiento de las actividades y revisión del trabajo de equipo son todas realizadas por la misma gente involucrada y además se hacen de manera manual. El mecanismo propuesto, por otro lado, se orienta hacia la automatización de los procesos y supone el uso de las tecnologías de información. En ese aspecto, la diferencia principal es la *forma de trabajo* hacia la que van dirigidos tanto el método propuesto por [24] como el mecanismo de monitoreo planteado en la presente investigación.

5.4. Contribuciones

A continuación, se explican brevemente las contribuciones de la presente investigación. Algunas de ellas fueron planteadas como aportaciones desde un inicio, y otras fueron obtenidas a lo largo del desarrollo.

Detección de discrepancias temporales en las actividades de un proyecto. El mecanismo cumple con la identificación de situaciones de origen temporal que salgan de lo establecido, y la detección de atrasos es completa. Ése era el objetivo central de la presente investigación, y se puede decir—tomando como referencia la conceptualización realizada y los resultados obtenidos mediante las pruebas—que el mecanismo basado en RT presentado lo satisface.

Uso del RT a través del Álgebra de Intervalos de Allen. Si bien el utilizar este tipo de razonamiento implica costos extra, pues se requieren hacer cálculos adicionales, también se puede argumentar que su uso trae ventajas en varios aspectos. Por ejemplo, el Razonamiento Temporal (representado en nuestro caso como el Álgebra de Intervalos de Allen) *per sé* resulta *flexible* en cuanto a que permite manipular los hechos dados, actualizarlos, propagar restricciones entre ellos (aún con relaciones bastante lejanas), etc. Este tipo de beneficios pueden ser provechosos si se explotan debidamente.

Integración de un módulo de notificación. Si bien las notificaciones que se generan actualmente son sólo locales, el mecanismo se encuentra preparado para el manejo de éstas, el cual es posible gracias a la integración a otros sistemas, tales como JITIK.

Orientación hacia automatización de procesos involucrados en el proyecto. Haciendo relación con el punto anterior, al estar pensado—y preparado—para adaptarse con sistemas multiagente, el mecanismo propuesto se encuentra *orientado* hacia la automatización de procesos, tales como el seguimiento y el control. Esto, en consecuencia, deja un camino abierto para que se continúe explorando el aspecto del uso de las TI (en específico de las TI relacionadas a la Inteligencia Artificial) para la administración de proyectos.

Apéndice A

Resultados Pruebas

A.1. Resultados arrojados en consola

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E1
     [java] CASO DE PRUEBA: c1
     [java] ********************************

     [java] Fecha actual: 0

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA<20>  0 20 0 20 -1 -1
     [java] Predecesores: START

     [java] ###################################

     [java] MILESTONE: 5

     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] LA ACTIVIDAD ALPHA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad ALPHA: NS
     [java] Estado que debiera tener: S

     [java] RESULTADO MONITOREO DE ALPHA: Existen discrepancias.
     [java] Tipo de discrepancia en ALPHA: 1

     [java] NOTIFICACION: Existe un atraso. Causa: Estados no coinciden.
     [java] Agentes responsables de la actividad: agent_alpha agent_project

     [java] Actividades relacionadas (a notificar):  Ninguna.
```

Figura A.1: Escenario 1, Caso 1

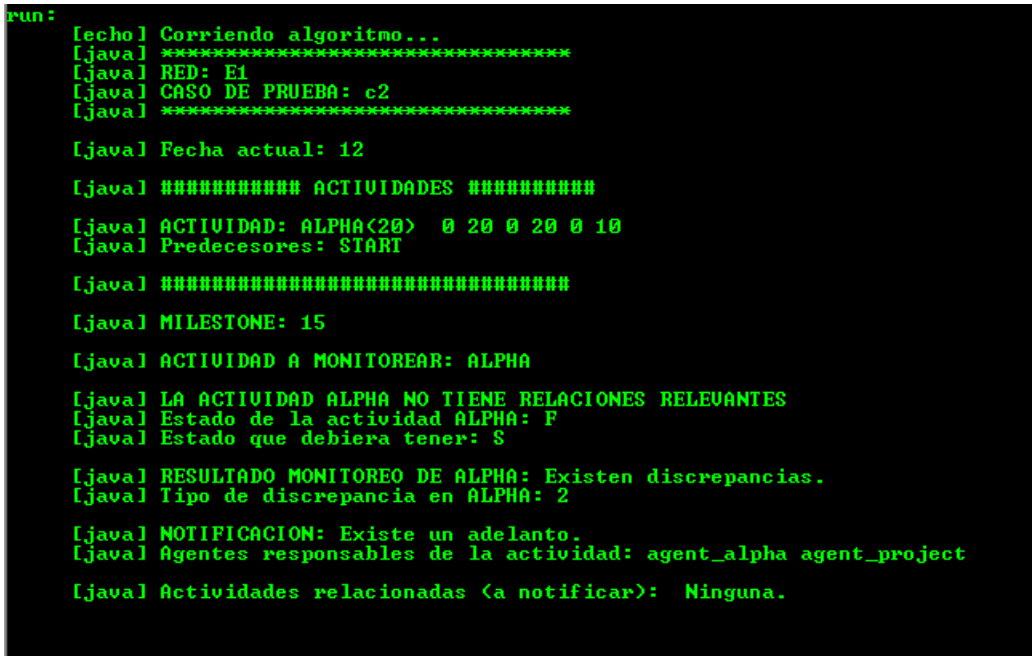

Figura A.2: Escenario 1, Caso 2

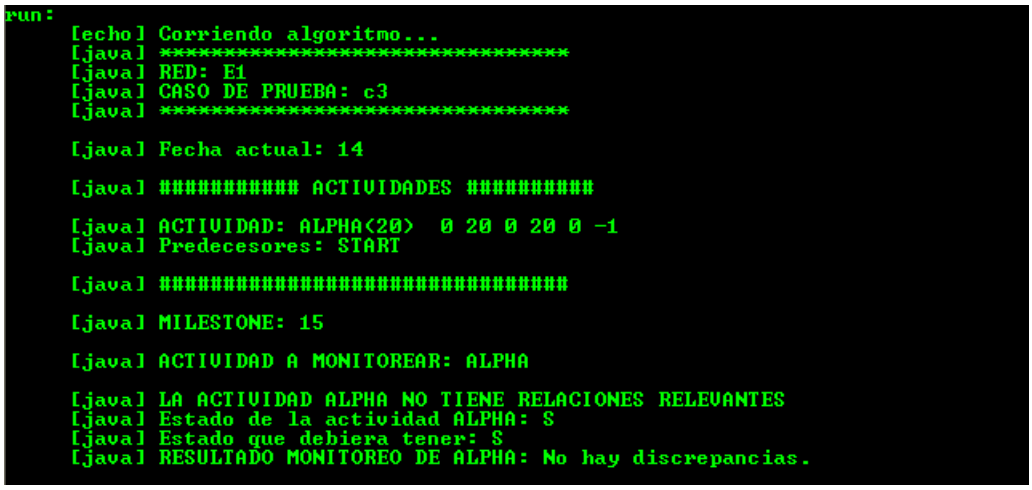

Figura A.3: Escenario 1, Caso 3

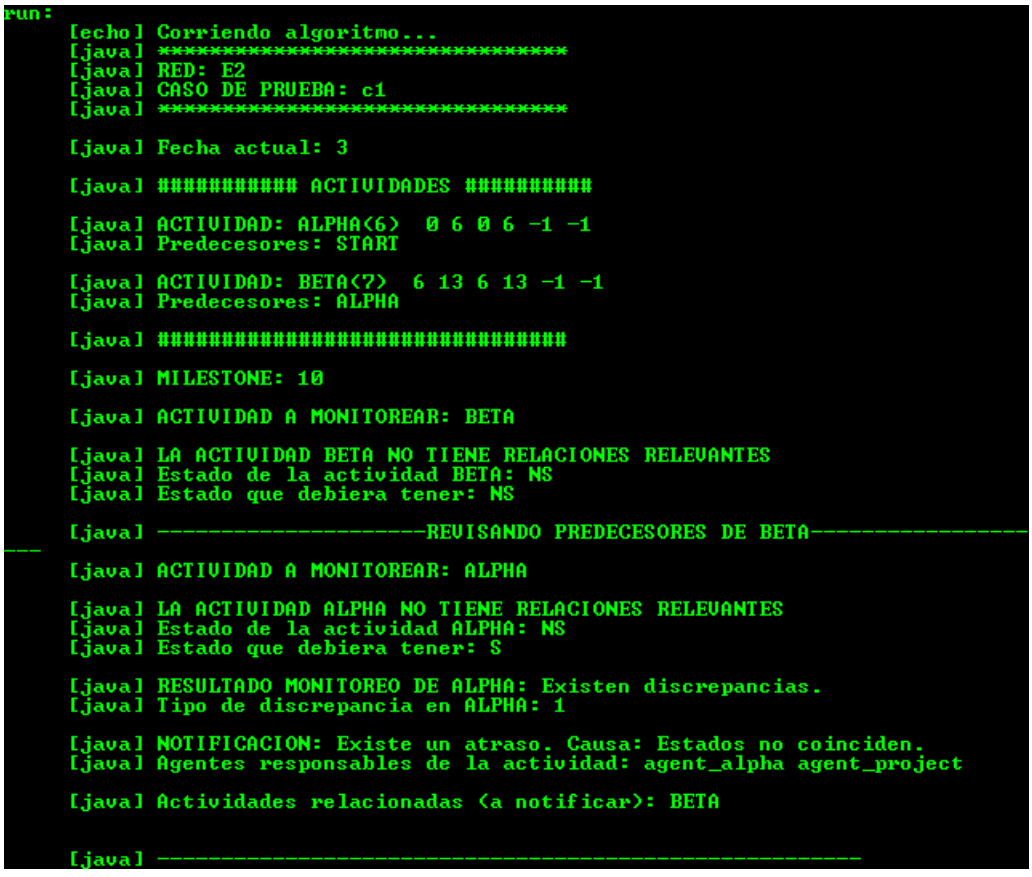

Figura A.4: Escenario 2, Caso 1

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E2
     [java] CASO DE PRUEBA: c2
     [java] ********************************

     [java] Fecha actual: 5

     [java] ############ ACTIVIDADES ##########

     [java] ACTIVIDAD: ALPHA(6)  0 6 0 6 0 4
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(7)  6 13 6 13 -1 -1
     [java] Predecesores: ALPHA

     [java] ###################################

     [java] MILESTONE: 10

     [java] ACTIVIDAD A MONITOREAR: BETA

     [java] LA ACTIVIDAD BETA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad BETA: NS
     [java] Estado que debiera tener: NS

     [java] ----------------------REVISANDO PREDECESORES DE BETA-----------------
---
     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] LA ACTIVIDAD ALPHA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad ALPHA: F
     [java] Estado que debiera tener: S

     [java] RESULTADO MONITOREO DE ALPHA: Existen discrepancias.
     [java] Tipo de discrepancia en ALPHA: 2

     [java] NOTIFICACION: Existe un adelanto.
     [java] Agentes responsables de la actividad: agent_alpha agent_project

     [java] Actividades relacionadas (a notificar): BETA


     [java] ------------------------------------------------------
```

Figura A.5: Escenario 2, Caso 2

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E2
     [java] CASO DE PRUEBA: c3
     [java] ********************************

     [java] Fecha actual: 9

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(6)  0 6 0 6 0 6
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(7)  6 13 6 13 -1 -1
     [java] Predecesores: ALPHA

     [java] ###################################

     [java] MILESTONE: 10

     [java] ACTIVIDAD A MONITOREAR: BETA

     [java] LA ACTIVIDAD BETA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad BETA: NS
     [java] Estado que debiera tener: S

     [java] RESULTADO MONITOREO DE BETA: Existen discrepancias.
     [java] Tipo de discrepancia en BETA: 1

     [java] NOTIFICACION: Existe un atraso. Causa: Estados no coinciden.
     [java] Agentes responsables de la actividad: agent_beta agent_project

     [java] Actividades relacionadas (a notificar):  Ninguna.
```

Figura A.6: Escenario 2, Caso 3

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E2
     [java] CASO DE PRUEBA: c4
     [java] ********************************

     [java] Fecha actual: 12

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(6)  0 6 0 6 0 6
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(7)  6 13 6 13 6 10
     [java] Predecesores: ALPHA

     [java] ###################################

     [java] MILESTONE: 10

     [java] ACTIVIDAD A MONITOREAR: BETA

     [java] LA ACTIVIDAD BETA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad BETA: F
     [java] Estado que debiera tener: S

     [java] RESULTADO MONITOREO DE BETA: Existen discrepancias.
     [java] Tipo de discrepancia en BETA: 2

     [java] NOTIFICACION: Existe un adelanto.
     [java] Agentes responsables de la actividad: agent_beta agent_project

     [java] Actividades relacionadas (a notificar):  Ninguna.
```

Figura A.7: Escenario 2, Caso 4

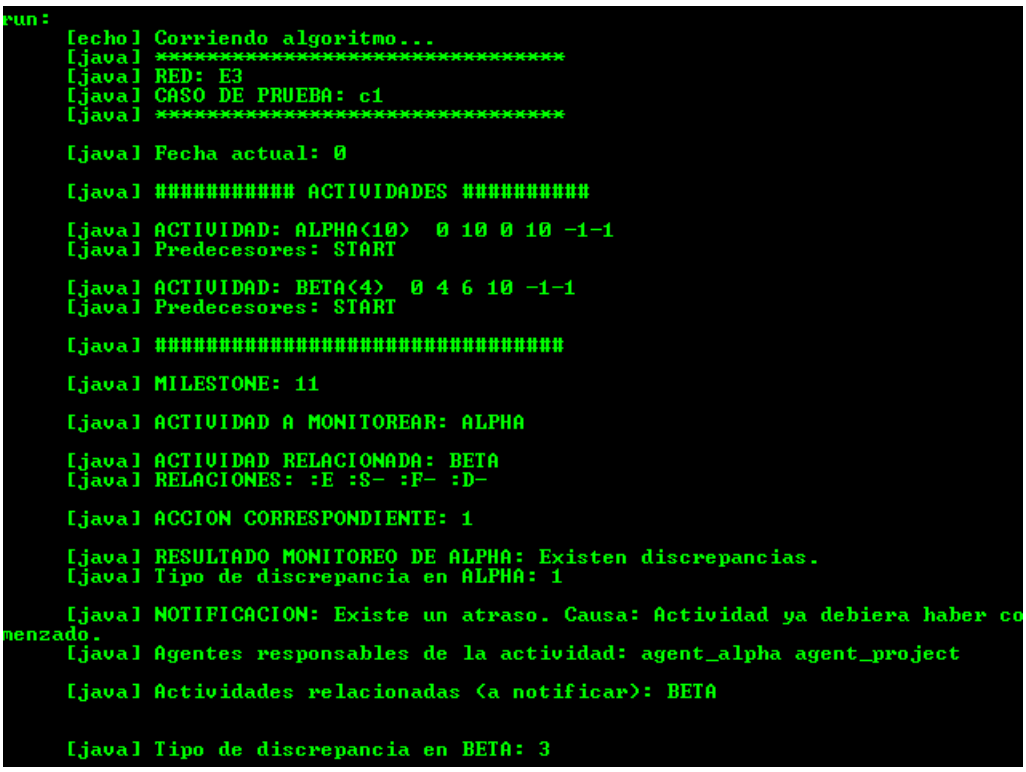

Figura A.8: Escenario 3, Caso 1

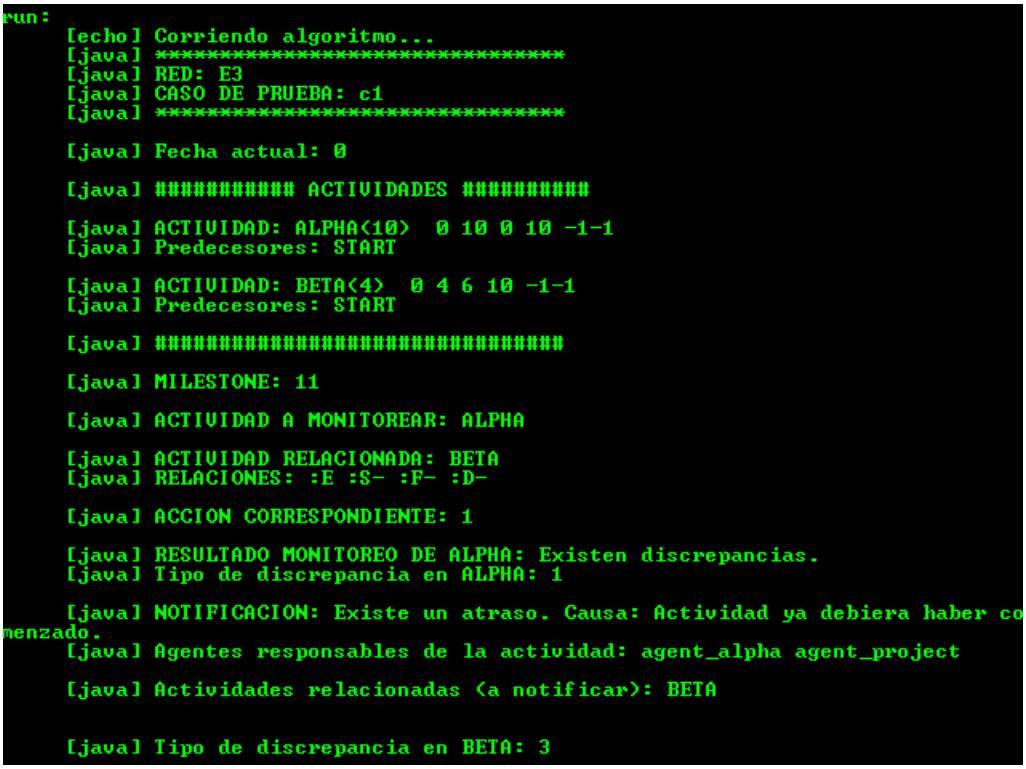

Figura A.9: Escenario 3, Caso 1

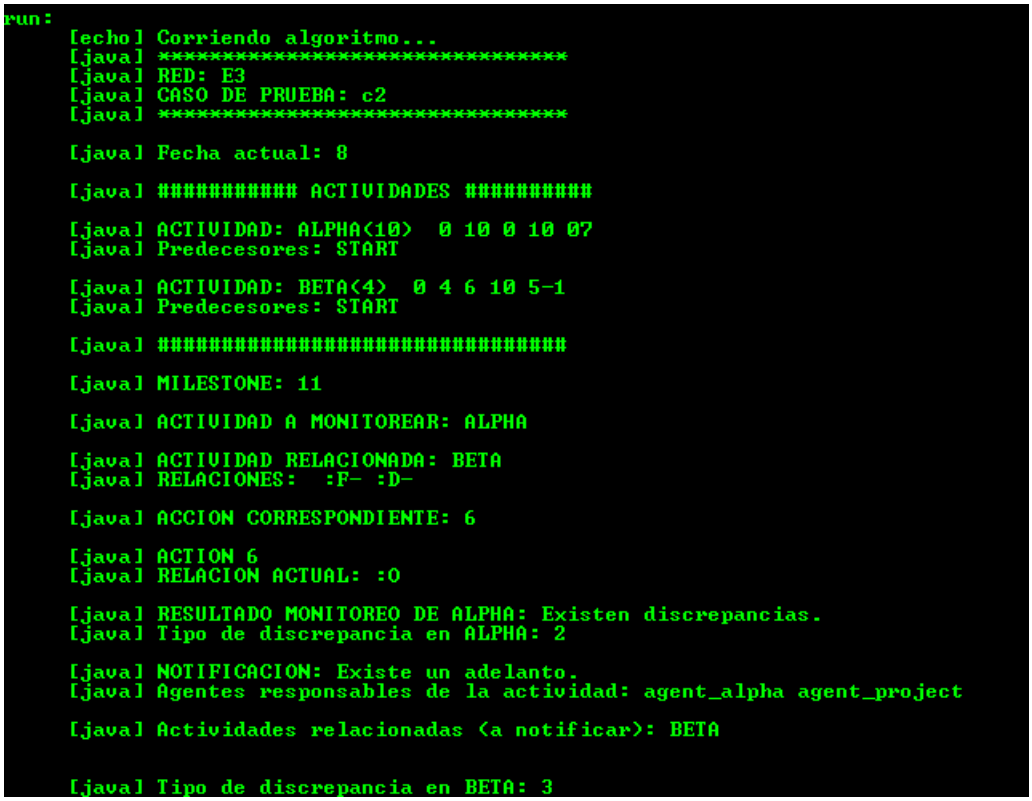

Figura A.10: Escenario 3, Caso 2

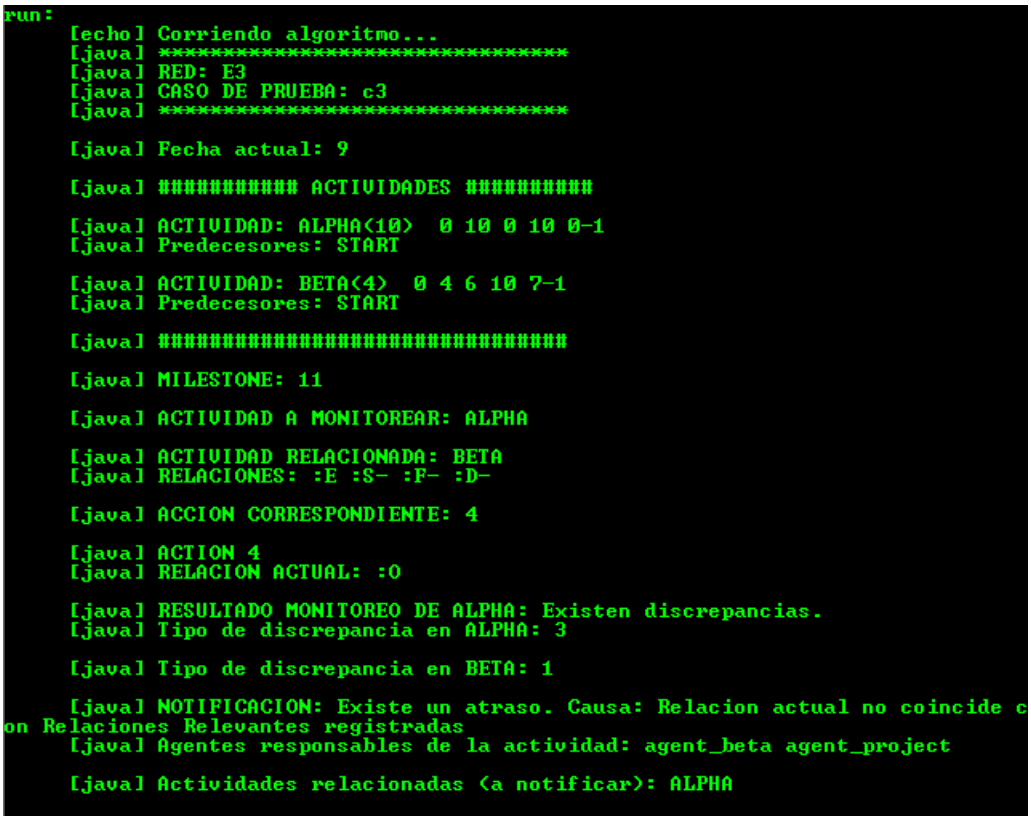

Figura A.11: Escenario 3, Caso 3

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E3
     [java] CASO DE PRUEBA: c4
     [java] ********************************

     [java] Fecha actual: 10

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(10)  0 10 0 10 0 10
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(4)  0 4 6 10 0 3
     [java] Predecesores: START

     [java] ###################################

     [java] MILESTONE: 11

     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] ACTIVIDAD RELACIONADA: BETA
     [java] RELACIONES: :E :S- :F- :D-

     [java] ACCION CORRESPONDIENTE: 9

     [java] RESULTADO MONITOREO DE ALPHA: Existen discrepancias.
     [java] Tipo de discrepancia en ALPHA: 3

     [java] Tipo de discrepancia en BETA: 2

     [java] NOTIFICACION: Existe un adelanto.
     [java] Agentes responsables de la actividad: agent_beta agent_project

     [java] Actividades relacionadas (a notificar): ALPHA
```

Figura A.12: Escenario 3, Caso 4

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E3
     [java] CASO DE PRUEBA: c4.v1
     [java] ********************************

     [java] Fecha actual: 7

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(10)  0 10 0 10 0 -1
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(4)  0 4 6 10 0 3
     [java] Predecesores: START

     [java] ###################################

     [java] MILESTONE: 11

     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] ACTIVIDAD RELACIONADA: BETA
     [java] RELACIONES: :E :S- :F- :D-

     [java] ACCION CORRESPONDIENTE: 5

     [java] ACTION 5
     [java] RELACION ACTUAL: :S-


     [java] RESULTADO MONITOREO DE ALPHA: No hay discrepancias.

     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: ALPHA BETA  :S-
     [java] Interseccion:  :S-
```

Figura A.13: Escenario 3, Caso 4, Variación 1

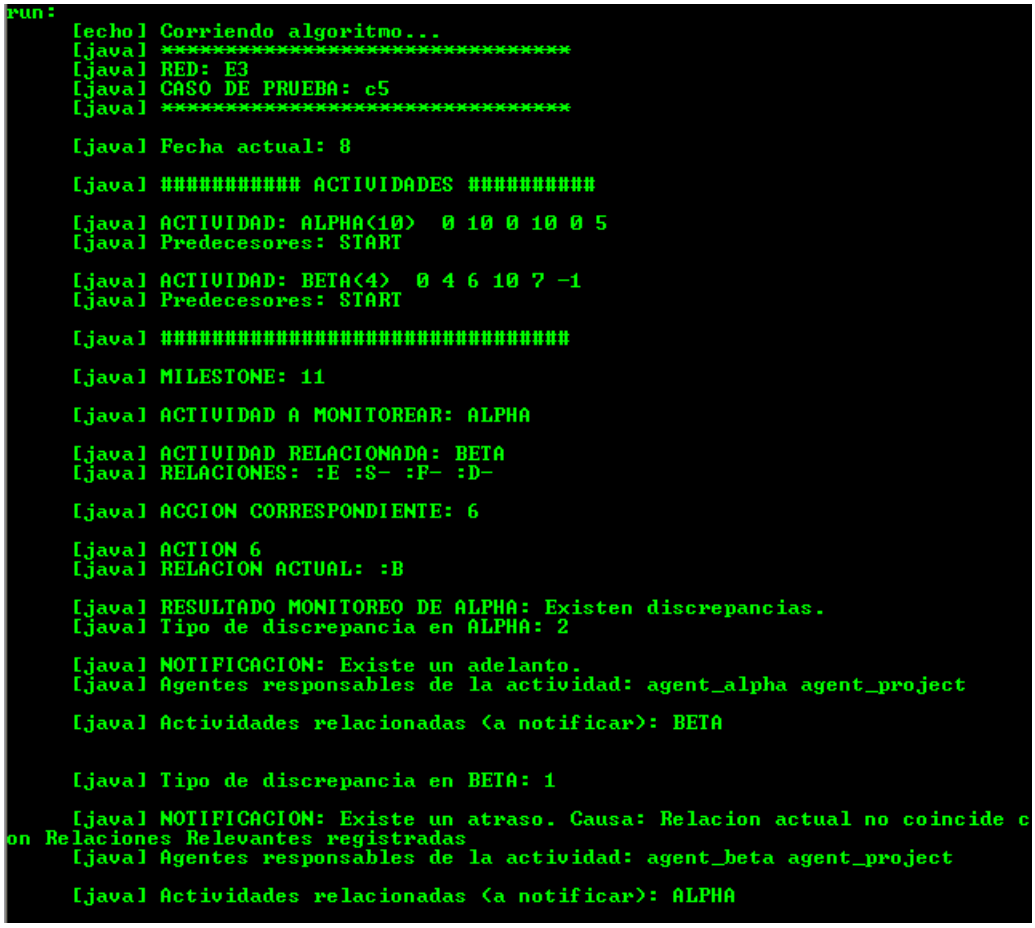

Figura A.14: Escenario 3, Caso 5

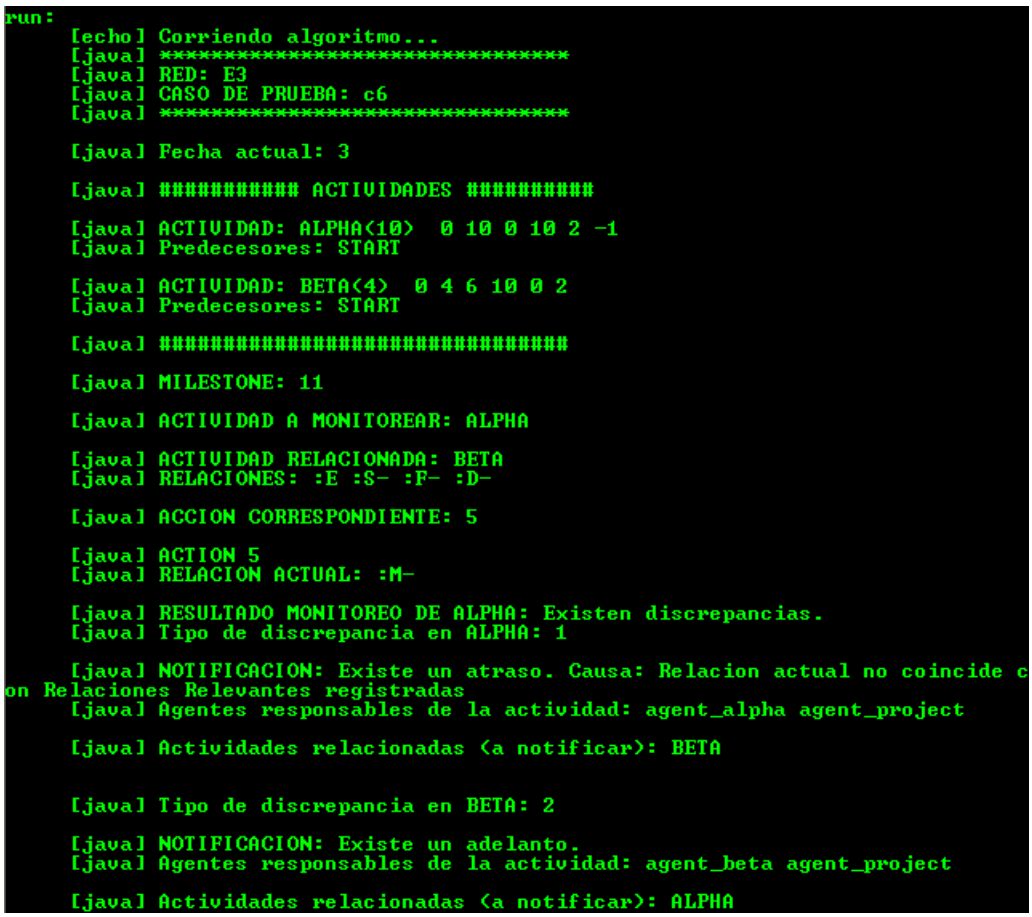

Figura A.15: Escenario 3, Caso 6

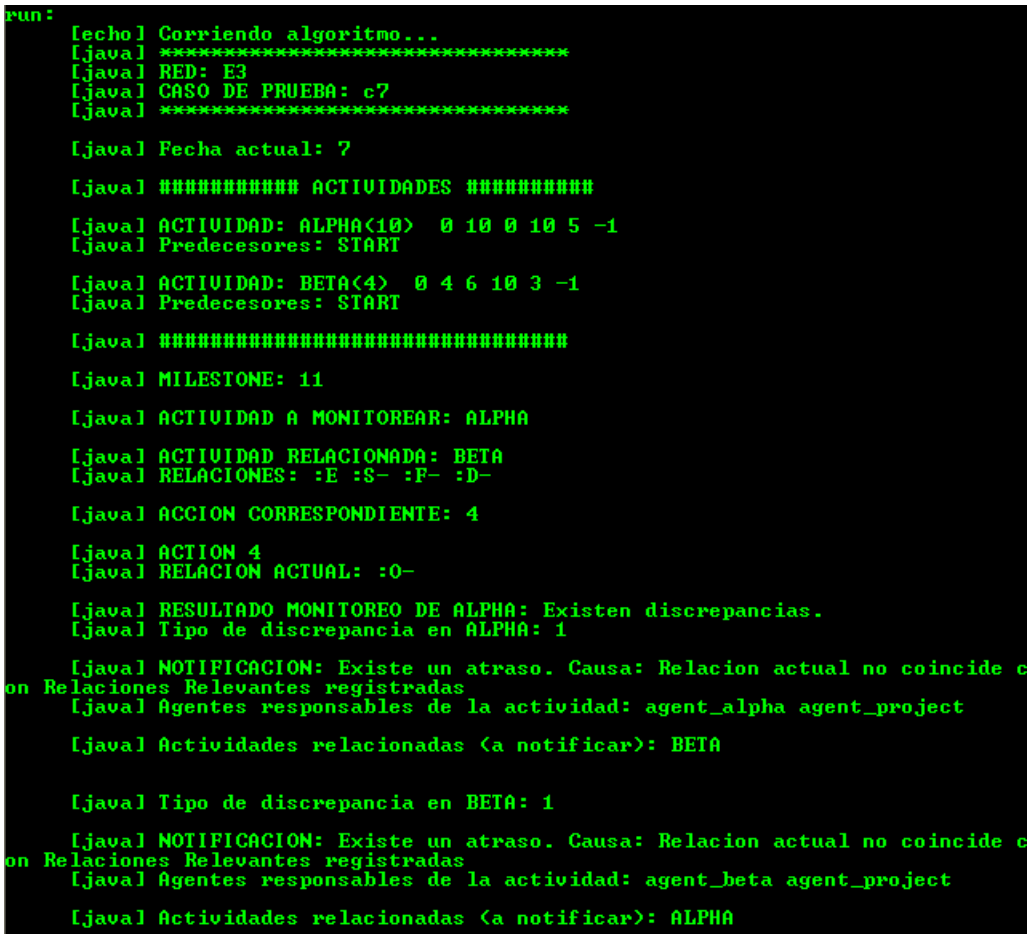

Figura A.16: Escenario 3, Caso 7

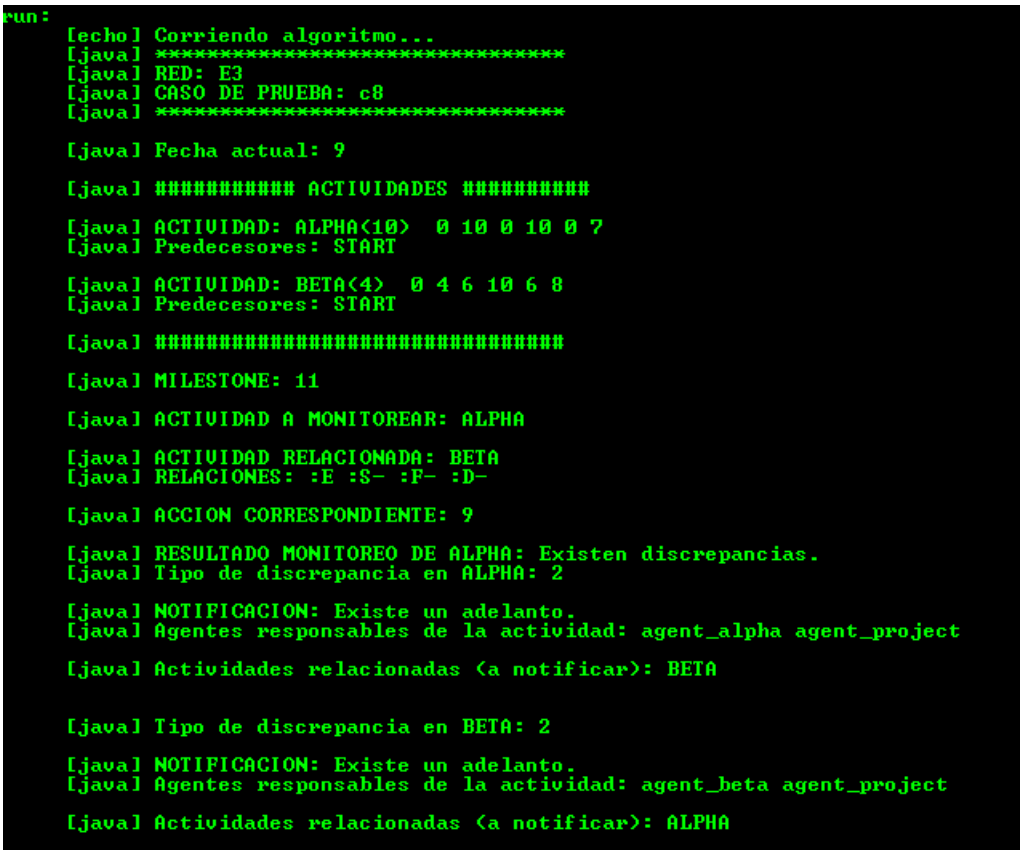

Figura A.17: Escenario 3, Caso 8

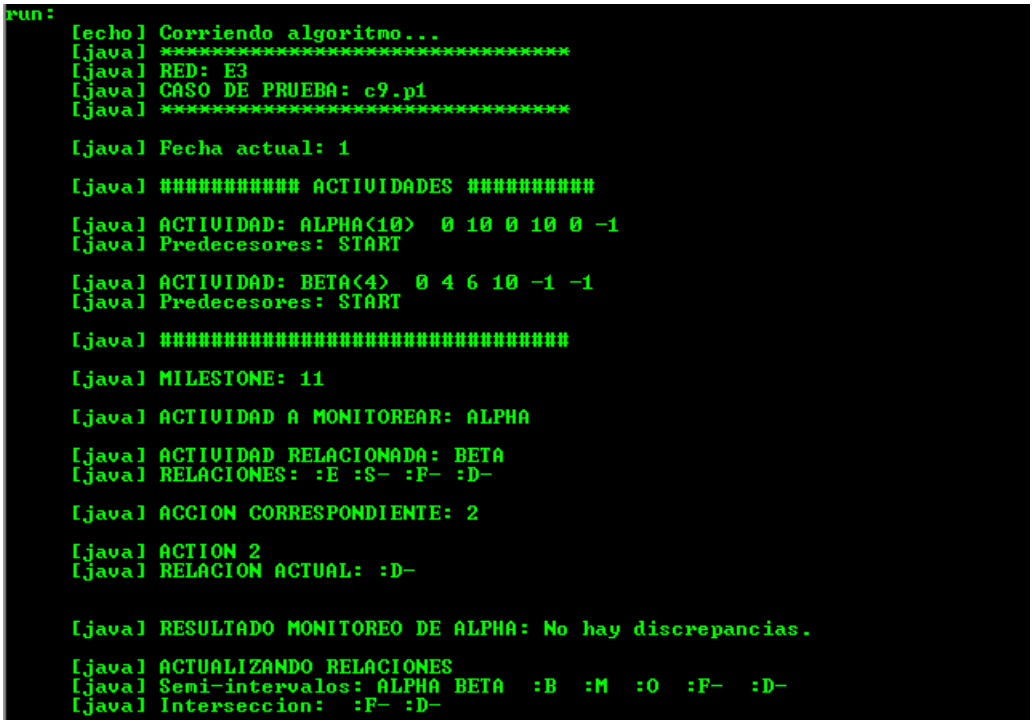

Figura A.18: Escenario 3, Caso 9, Parte 1

```
run:
     [echo] Corriendo algoritmo...
     [java] **********************************
     [java] RED: E3
     [java] CASO DE PRUEBA: c9.p2
     [java] **********************************

     [java] Fecha actual: 3

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(10)  0 10 0 10 0 -1
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(4)  0 4 6 10 3 -1
     [java] Predecesores: START

     [java] ###################################

     [java] MILESTONE: 11

     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] ACTIVIDAD RELACIONADA: BETA
     [java] RELACIONES:  :F- :D-

     [java] ACCION CORRESPONDIENTE: 4

     [java] ACTION 4
     [java] RELACION ACTUAL: :D-


     [java] RESULTADO MONITOREO DE ALPHA: No hay discrepancias.

     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: ALPHA BETA  :O  :F-  :D-
     [java] Interseccion:  :F- :D-
```

Figura A.19: Escenario 3, Caso 9, Parte 2

```
run:
     [echo] Corriendo algoritmo...
     [java] **********************************
     [java] RED: E3
     [java] CASO DE PRUEBA: c9.p3
     [java] **********************************

     [java] Fecha actual: 8

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(10)  0 10 0 10 0 -1
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(4)  0 4 6 10 3 7
     [java] Predecesores: START

     [java] ###################################

     [java] MILESTONE: 11

     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] ACTIVIDAD RELACIONADA: BETA
     [java] RELACIONES:  :F- :D-

     [java] ACCION CORRESPONDIENTE: 5

     [java] ACTION 5
     [java] RELACION ACTUAL: :D-


     [java] RESULTADO MONITOREO DE ALPHA: No hay discrepancias.

     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: ALPHA BETA  :D-
     [java] Interseccion:  :D-
```

Figura A.20: Escenario 3, Caso 9, Parte 3

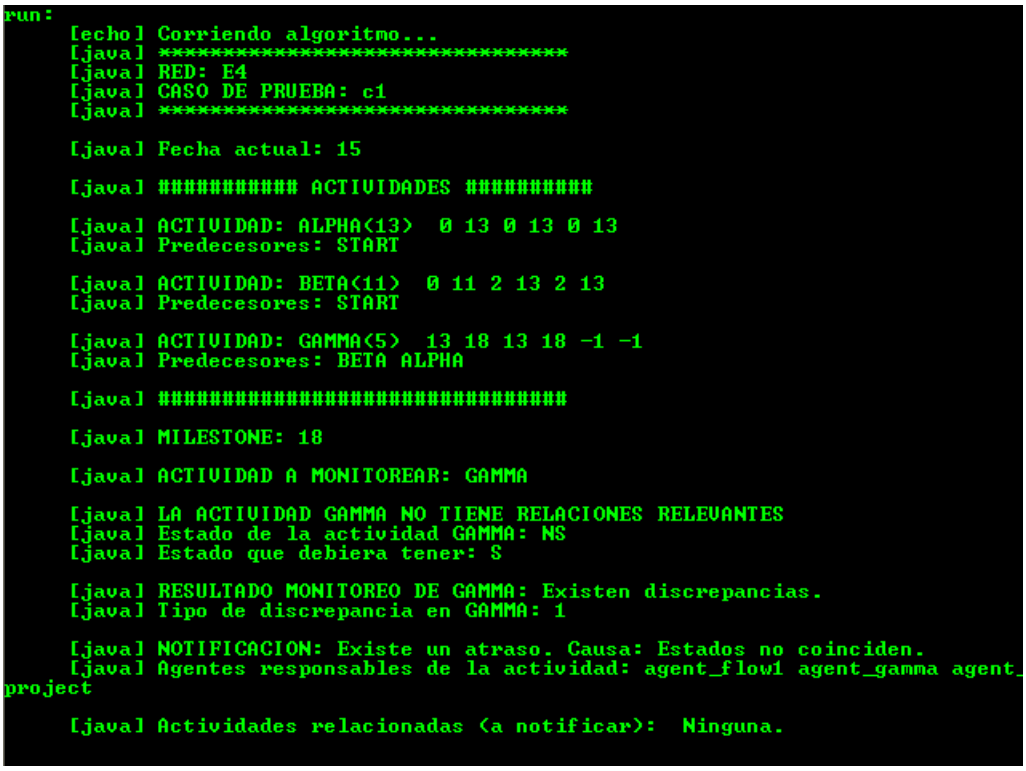

Figura A.21: Escenario 4, Caso 1, Parte 1

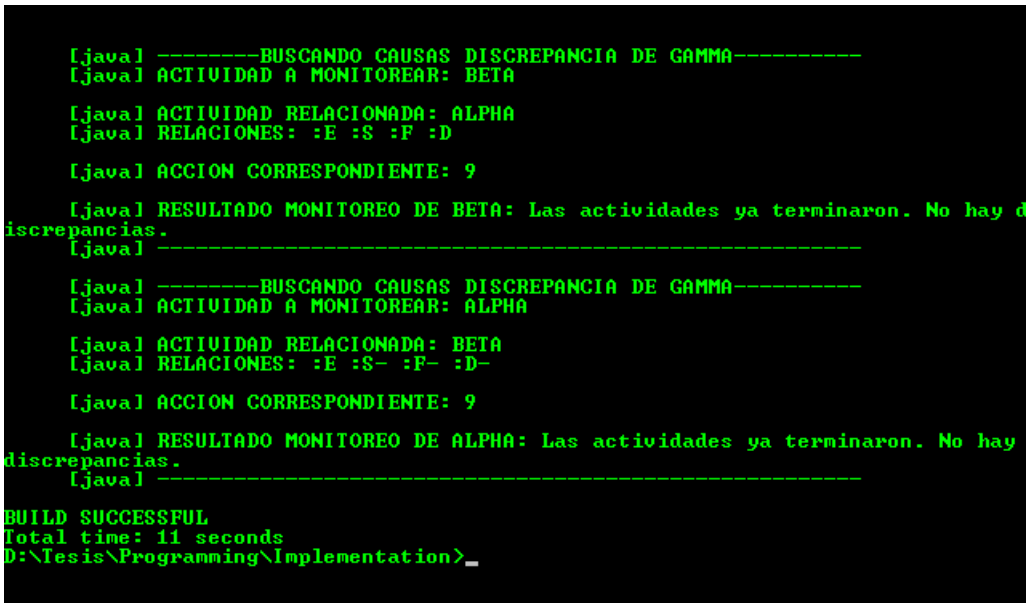

Figura A.22: Escenario 4, Caso 1, Parte 2

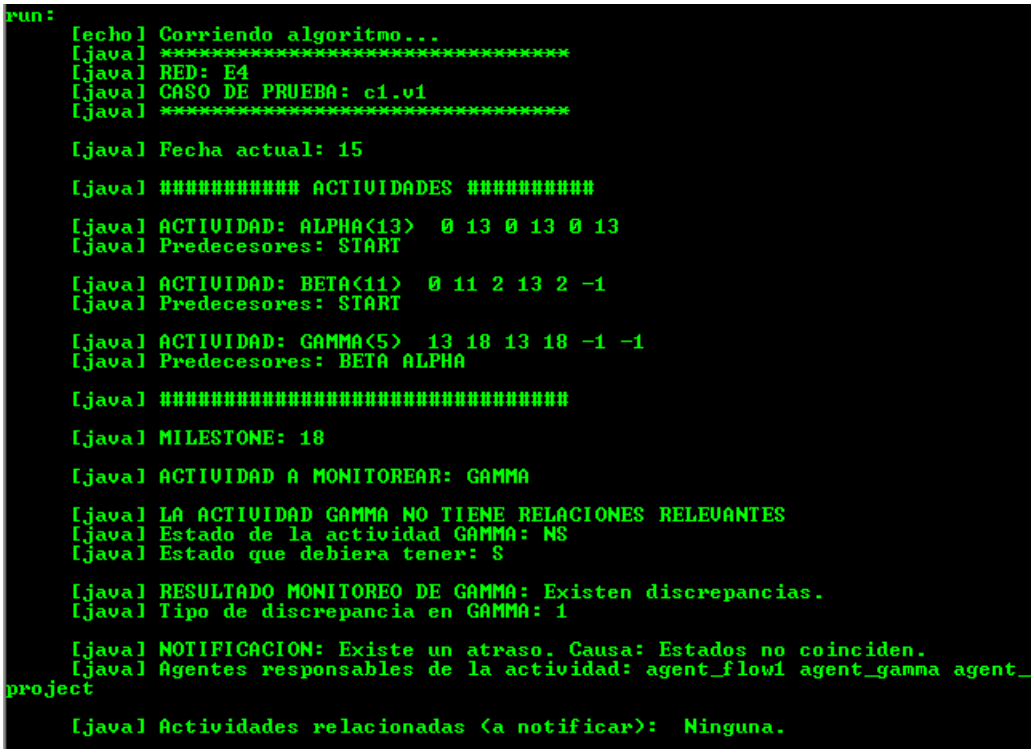

Figura A.23: Escenario 4, Caso 1, Variación 1, Parte 1

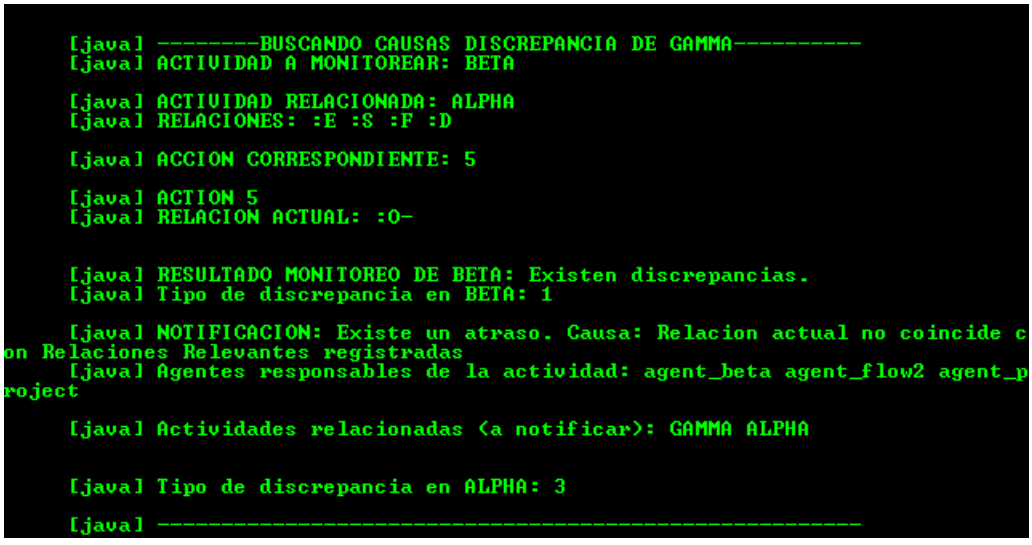

Figura A.24: Escenario 4, Caso 1, Variación 1, Parte 2

```
run:
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E4
     [java] CASO DE PRUEBA: c1.v2
     [java] ********************************

     [java] Fecha actual: 15

     [java] ########### ACTIVIDADES ##########

     [java] ACTIVIDAD: ALPHA(13)  0 13 0 13 0 -1
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(11)  0 11 2 13 2 13
     [java] Predecesores: START

     [java] ACTIVIDAD: GAMMA(5)  13 18 13 18 -1 -1
     [java] Predecesores: BETA ALPHA

     [java] ###################################

     [java] MILESTONE: 18

     [java] ACTIVIDAD A MONITOREAR: GAMMA

     [java] LA ACTIVIDAD GAMMA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad GAMMA: NS
     [java] Estado que debiera tener: S

     [java] RESULTADO MONITOREO DE GAMMA: Existen discrepancias.
     [java] Tipo de discrepancia en GAMMA: 1

     [java] NOTIFICACION: Existe un atraso. Causa: Estados no coinciden.
     [java] Agentes responsables de la actividad: agent_flow1 agent_gamma agent_
project

     [java] Actividades relacionadas (a notificar):  Ninguna.
```

Figura A.25: Escenario 4, Caso 1, Variación 2, Parte 1

```
     [java] --------BUSCANDO CAUSAS DISCREPANCIA DE GAMMA----------
     [java] ACTIVIDAD A MONITOREAR: BETA

     [java] ACTIVIDAD RELACIONADA: ALPHA
     [java] RELACIONES: :E :S :F :D

     [java] ACCION CORRESPONDIENTE: 6

     [java] ACTION 6
     [java] RELACION ACTUAL: :D


     [java] RESULTADO MONITOREO DE BETA: No hay discrepancias.

     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: BETA ALPHA  :D
     [java] Interseccion:  :D
     [java] ----------------------------------------------------

     [java] --------BUSCANDO CAUSAS DISCREPANCIA DE GAMMA----------
     [java] ACTIVIDAD A MONITOREAR: ALPHA

     [java] ACTIVIDAD RELACIONADA: BETA
     [java] RELACIONES: :E :S- :F- :D-

     [java] ACCION CORRESPONDIENTE: 5

     [java] ACTION 5
     [java] RELACION ACTUAL: :D-


     [java] RESULTADO MONITOREO DE ALPHA: No hay discrepancias.

     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: ALPHA BETA  :D-
     [java] Interseccion:  :D-
     [java] ----------------------------------------------------
```

Figura A.26: Escenario 4, Caso 1, Variación 2, Parte 2

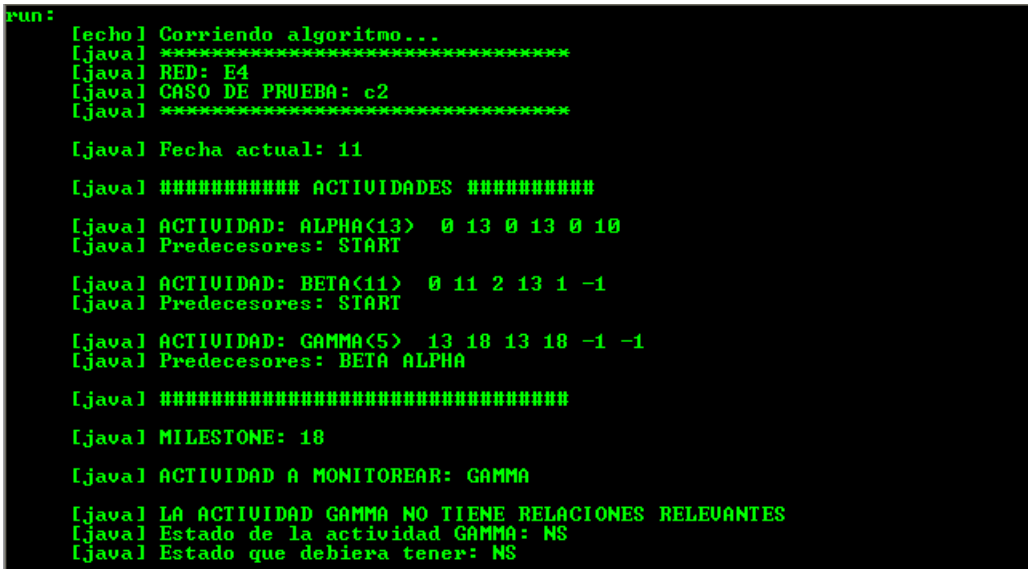

Figura A.27: Escenario 4, Caso 2, Parte 1

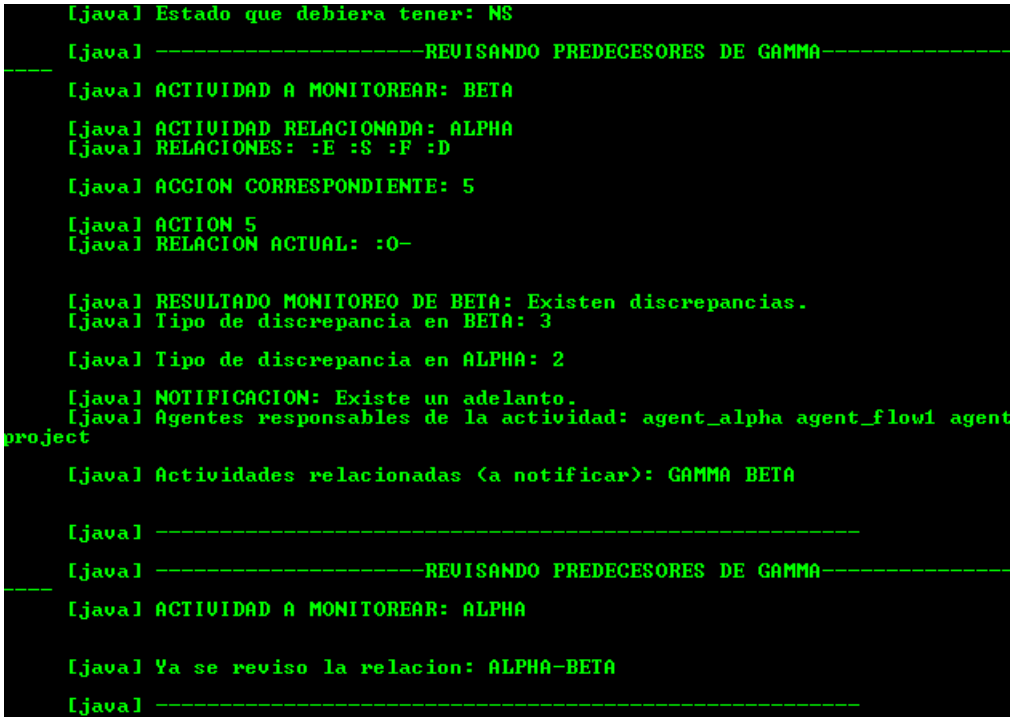

Figura A.28: Escenario 4, Caso 2, Parte 2

```
     [echo] Corriendo algoritmo...
     [java] ********************************
     [java] RED: E4
     [java] CASO DE PRUEBA: c3
     [java] ********************************

     [java] Fecha actual: 8

     [java] ############ ACTIVIDADES ###########

     [java] ACTIVIDAD: ALPHA(13)  0 13 0 13 0 -1
     [java] Predecesores: START

     [java] ACTIVIDAD: BETA(11)  0 11 2 13 0 7
     [java] Predecesores: START

     [java] ACTIVIDAD: GAMMA(5)  13 18 13 18 -1 -1
     [java] Predecesores: BETA ALPHA

     [java] ###################################

     [java] MILESTONE: 18

     [java] ACTIVIDAD A MONITOREAR: GAMMA

     [java] LA ACTIVIDAD GAMMA NO TIENE RELACIONES RELEVANTES
     [java] Estado de la actividad GAMMA: NS
     [java] Estado que debiera tener: NS

     [java] ---------------------REVISANDO PREDECESORES DE GAMMA----------------
---
     [java] ACTIVIDAD A MONITOREAR: BETA

     [java] ACTIVIDAD RELACIONADA: ALPHA
     [java] RELACIONES: :E :S :F :D

     [java] ACCION CORRESPONDIENTE: 6

     [java] ACTION 6
     [java] RELACION ACTUAL: :S


     [java] RESULTADO MONITOREO DE BETA: No hay discrepancias.

     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: BETA ALPHA  :S
     [java] Interseccion:  :S
     [java] ACTUALIZANDO RELACIONES
     [java] Semi-intervalos: ALPHA BETA  :S-
     [java] Interseccion:  :S-
     [java] ----------------------------------------------------

     [java] ---------------------REVISANDO PREDECESORES DE GAMMA----------------
---
     [java] ACTIVIDAD A MONITOREAR: ALPHA


     [java] Ya se reviso la relacion: ALPHA-BETA
```

Figura A.29: Escenario 4, Caso 3

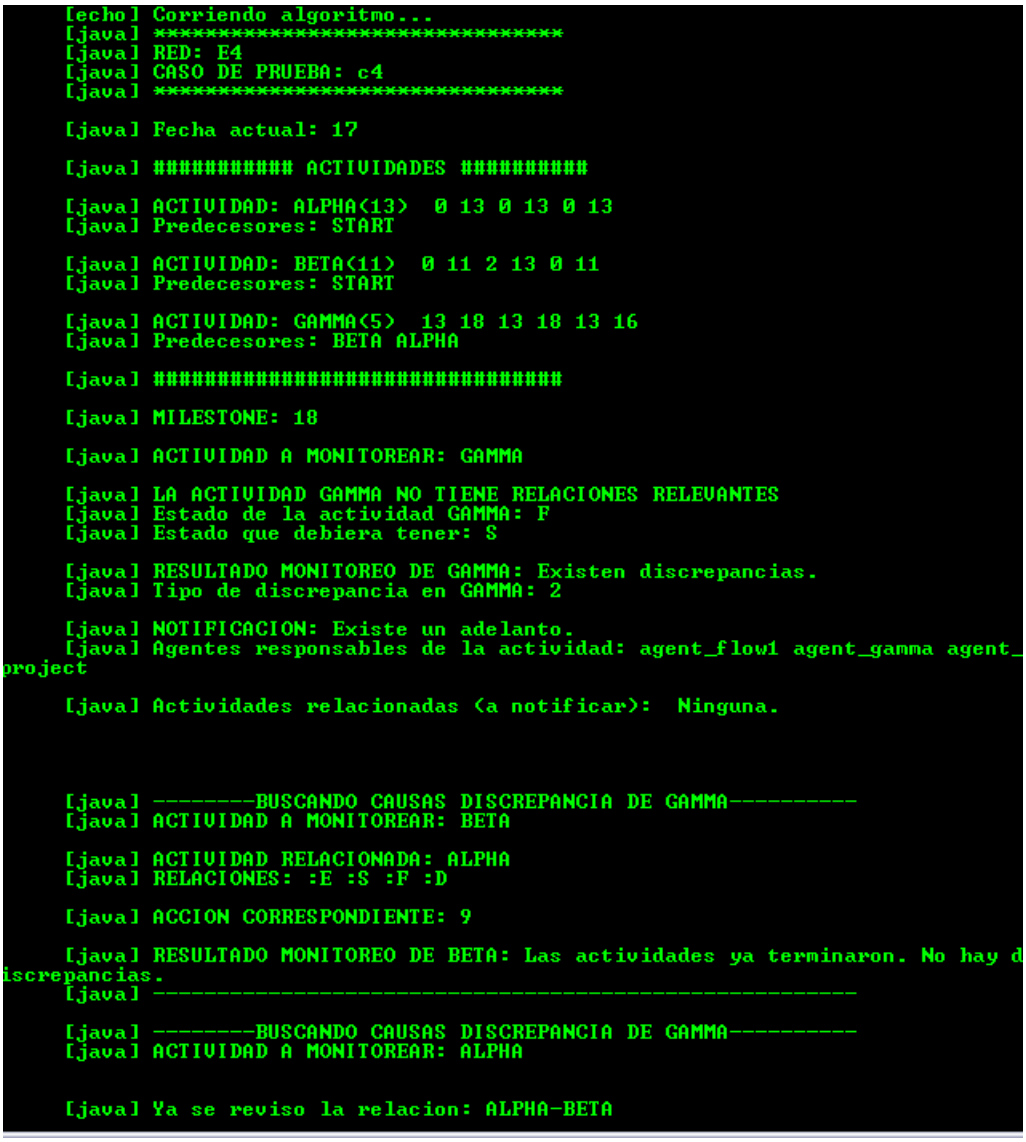

Figura A.30: Escenario 4, Caso 4

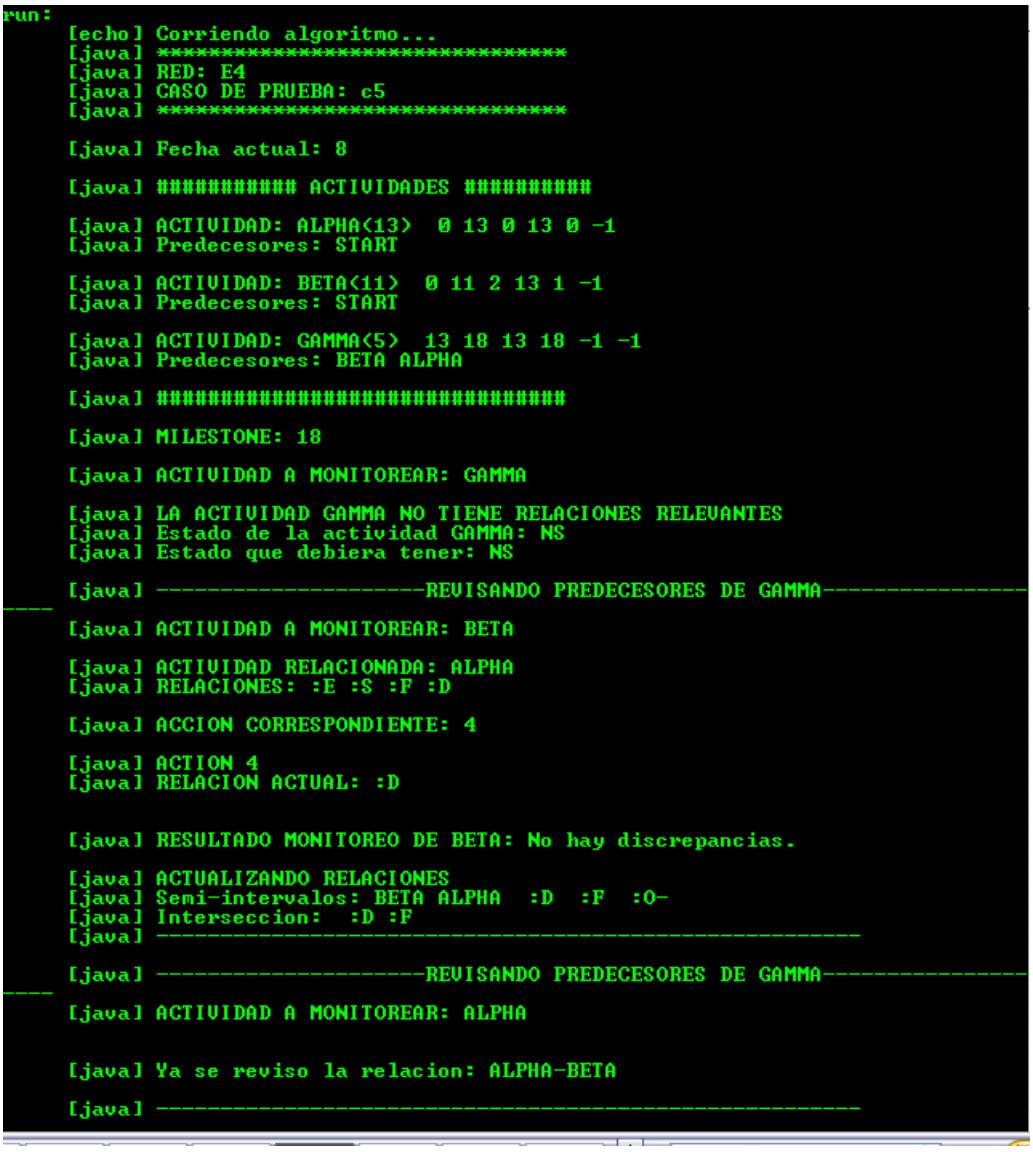

Figura A.31: Escenario 4, Caso 5

Bibliografía

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 1983.

[2] James F. Allen. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems*, 1991.

[3] J.C. Augusto, C.D. Nugent, and N.D. Black. Management and analysis of time-related data in smart home environment. Technical report, University of Ulster at Jordanstown, 2004.

[4] R. Brena, J. Aguirre, and A.C. Treviño. Just-in-Time Information and Knowledge: Agent technology for KM Bussiness Process. In *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference*, 2001.

[5] David Brojt. *Project Management: Un enfoque y ejecución de proyectos en la empresa para aplicar el lunes por la mañana*. Granica, Florida, 2005.

[6] David I. Cleland. *Project Management: Strategic Design and Implementation*. McGraw-Hill, New York, 1999.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stern. *An Introduction to Algorithms*. McGraw-Hill, Massachusetts, 2003.

[8] Manuel de Cos Castillo. *Teoría General del Proyecto: Dirección de Proyectos*. Editorial Síntesis, Madrid, 1995.

[9] Salah E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. John Wiley and Sons, Inc., New York, 1977.

[10] Shimon Even. *Graph Algorithms*. Computer Science Press, Maryland, 1979.

[11] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 1992.

[12] Alfonso Gerevini, Lenhart Schubert, and Stephanie Shcaeffer. The temporal reasoning systems timegraph i-ii. Technical report, Istituto per la Ricerca Scientifica e Tecnologica, University of Rochester, University of Alberta, 1995.

[13] Ralph P. Grimaldi. *Matemáticas Discreta y Combinatoria*. Addison Wesley Longman, Massachusetts, 1998.

[14] ITESM CSI. *Just In Time Information and Knowledge*, 2006. (`http://lizt.mty.itesm.mx/jitik`).

[15] Kenneth E. Kendall and Julie E. Kendall. *Systems Analysis and Design*. Prentice Hall, New Jersey, 1999.

[16] Ted Klastorin. *Administración de Proyectos*. Alfaomega, México, 2006.

[17] Johannes A.G.M. Koomen. The timelogic temporal reasoning system. Technical report, University of Rochester, 1989.

[18] Kenneth C. Laudon and Jane P. Laudon. *Management Information Systems: Managing the Digital Firm*. Prentice Hall, New Jersey, 2004.

[19] Peter W. G. Morris and Jeffrey K. Pinto. *The Wiley Guide to Managing Projects*. John Wiley and Sons, Inc., New Jersey, 2004.

[20] Bernard A. Nadel. Some applications of the constraint satisfaction problem. Technical report, Wayne State University, June 1990.

[21] John M. Nicholas. *Project Management for Business and Technology: Principles and Practice*. Prentice Hall, New Jersey, 2001.

[22] Martin O'Connor, William E. Grosso, Samson W. Tu, and Mark A. Musen. Rasta: A distributed temporal abstraction system to facilitate knowledge-driven monitoring of clinical databases. Technical report, Stanford School of Medicine, 1995.

[23] PMI. *A Guide to the Project Management Body of Knowledge*. Project Management Institute, Pennsylvania, 2004.

[24] Marc Rettig and Gary Simons. A project planning and development process for small teams. *Communications of the ACM*, 1993.

[25] Stuart Russell and Peter Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, New Jersey, 2003.

[26] Yuval Shahar. A framework for knowledge-based temporal abstraction. Technical report, Stanford University, June 1997.

[27] Avraham Shtub, Jonathan F. Bard, and Shlomo Globerson. *Project Management: Engineering, Technology, and Implementation*. Prentice Hall, New Jersey, 1994.

[28] Lluís Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7, March 1994.

[29] Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the AAAI-86*, 1986.