Universidad de Alcalá
COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

ACTA DE EVALUACIÓN DE LA TESIS DOCTORAL

Año académico 2016/17

DOCTORANDO: **ALLUÉ GIL, ALBERTO**
D.N.I./PASAPORTE: ******7972L**

PROGRAMA DE DOCTORADO: **D338 CIENCIAS DE COMPUTACIÓN**
DEPARTAMENTO DE: **CIENCIAS DE LA COMPUTACIÓN**
TITULACIÓN DE DOCTOR EN: **DOCTOR/A POR LA UNIVERSIDAD DE ALCALÁ**

En el día de hoy 04/09/17, reunido el tribunal de evaluación nombrado por la Comisión de Estudios Oficiales de Posgrado y Doctorado de la Universidad y constituido por los miembros que suscriben la presente Acta, el aspirante defendió su Tesis Doctoral, elaborada bajo la dirección de ELADIO DOMÍNGUEZ MURILLO // MARÍA ANTONIA ZAPATA ABAD.

Sobre el siguiente tema: *MODELO DE ACONTECIMIENTOS PARA LA PERSISTENCIA*

Finalizada la defensa y discusión de la tesis, el tribunal acordó otorgar la CALIFICACIÓN GLOBAL[1] de (**no apto, aprobado, notable** y **sobresaliente**): SOBRESALIENTE

Alcalá de Henares, 4 de septiembre de 2017

EL PRESIDENTE

EL SECRETARIO

EL VOCAL

Fdo.: José Antonio Gutiérrez de Mesa Fdo.: Jorge Lloret Gato Fdo.: Laureano Lambán Pardo

FIRMA DEL ALUMNO

Fdo.: .....

Con fecha 14 de septiembre de 2017 la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado, a la vista de los votos emitidos de manera anónima por el tribunal que ha juzgado la tesis, resuelve:

☒ Conceder la Mención de "Cum Laude"
☐ No conceder la Mención de "Cum Laude"

La Secretaria de la Comisión Delegada

[1] La calificación podrá ser "no apto" "aprobado" "notable" y "sobresaliente". El tribunal podrá otorgar la mención de "cum laude" si la calificación global es de sobresaliente y se emite en tal sentido el voto secreto positivo por unanimidad.

INCIDENCIAS / OBSERVACIONES:

Universidad de Alcalá
COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

En aplicación del art. 14.7 del RD. 99/2011 y el art. 14 del Reglamento de Elaboración, Autorización y Defensa de la Tesis Doctoral, la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado y Doctorado, en sesión pública de fecha 14 de septiembre, procedió al escrutinio de los votos emitidos por los miembros del tribunal de la tesis defendida por *ALLUÉ GIL, ALBERTO*, el día 4 de septiembre de 2017, titulada *MODELO DE ACONTECIMIENTOS PARA LA PERSISTENCIA*, para determinar, si a la misma, se le concede la mención "cum laude", arrojando como resultado el voto favorable de todos los miembros del tribunal.

Por lo tanto, la Comisión de Estudios Oficiales de Posgrado **resuelve otorgar** a dicha tesis la

MENCIÓN "CUM LAUDE"

Alcalá de Henares, 21 de septiembre de 2017
EL PRESIDENTE DE LA COMISIÓN DE ESTUDIOS
OFICIALES DE POSGRADO Y DOCTORADO

Firmado digitalmente por
VELASCO PEREZ JUAN
RAMON - DNI 03087239H
Fecha: 2017.09.22 11:23:49
+02'00'

Juan Ramón Velasco Pérez

Copia por e-mail a:
Doctorando: ALLUÉ GIL, ALBERTO
Secretario del Tribunal: JORGE LLORET GAZO
Directores de Tesis: ELADIO DOMÍNGUEZ MURILLO // MARÍA ANTONIA ZAPATA ABAD

Universidad de Alcalá

ESCUELA DE DOCTORADO
Servicio de Estudios Oficiales de Posgrado

DILIGENCIA DE DEPÓSITO DE TESIS.

Comprobado que el expediente académico de D./Dª ALBERTO ALLUÉ GIL reúne los requisitos exigidos para la presentación de la Tesis, de acuerdo a la normativa vigente, y habiendo presentado la misma en formato: ☐ soporte electrónico ☒ impreso en papel, para el depósito de la misma, en el Servicio de Estudios Oficiales de Posgrado, con el nº de páginas: 232 se procede, con fecha de hoy a registrar el depósito de la tesis.

Alcalá de Henares a 26 de abril de 2017

Emilia Rivero Antúnez

Fdo. El Funcionario

Universidad de Alcalá

DPTO. DE CIENCIAS DE LA COMPUTACIÓN
Escuela Politécnica Superior
Campus Universitario. Edificio Politécnico
28871 Alcalá de Henares (Madrid)
Teléfonos: 91 885 6645 Fax: 91 885 6646
e-mail: dpto.ccomputacion@uah.es

Dr. D. Eladio Domínguez Murillo, Catedrático de Universidad del Área de Ciencia de la Computación e Inteligencia Artificial del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza,

Dr. Dña. María Antonia Zapata Abad, Profesor Titular de Universidad del Área de Ciencia de la Computación e Inteligencia Artificial del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza,

CERTIFICAN: Que una vez concluido el trabajo de Tesis Doctoral titulado "Modelo de Acontecimientos para la Persistencia" realizado por D. Alberto Allué Gil, dicho trabajo tiene suficientes méritos teóricos, que se han contrastado adecuadamente mediante validaciones experimentales y que son altamente novedosos. Por todo ello se considera que procede su defensa pública.

Y para que conste, firman la presente en Zaragoza, a 5 de abril de 2017

Los Directores de la Tesis

Fdo.: Dr. Eladio Domínguez Murillo

Fdo.: Dr. Mª Antonia Zapata Abad

Universidad de Alcalá

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
Escuela Politécnica Superior
Campus Universitario. Edificio Politécnico
28871 Alcalá de Henares (Madrid)
Teléfonos: +34 91 885 66 51
Fax: +34 91 885 6646

Dra. Dña. Teresa I. Díez Folledo, Profesora Titular de Universidad del Área de Lenguajes y Sistemas Informáticos, en calidad de Directora del Departamento de Ciencias de la Computación.

CERTIFICO: Que una vez concluido el trabajo de la Tesis Doctoral titulada **"Modelo de Acontecimientos para la Persistencia"** realizada por D. Alberto Allué Gil y dirigida por los Drs. D. Eladio Dominguez Murillo y Dª. Maria Antonia Zapata Abad, reúne los requisitos para su presentación y defensa pública.

Y para que así conste, firmo la presente en Alcalá de Henares, a 21 de Abril de 2017

La Directora del Departamento de Ciencias de la Computación

Dra. Dña. Teresa Díez Folledo

Universidad de Alcalá

Programa de Doctorado en Ingeniería de la Información y del Conocimiento

Modelo de Acontecimientos para la Persistencia

Tesis doctoral presentada por

Alberto Allué Gil

2017

Programa de Doctorado en Ingeniería de la Información y del Conocimiento

Modelo de Acontecimientos para la Persistencia

Tesis doctoral presentada por

Alberto Allué Gil

Directores:
Dr. Eladio Domínguez Murillo
Dra. María Antonia Zapata Abad

Alcalá de Henares, 2017

A mi mujer.

A mi hijo.

Agradecimientos

Quiero agradecer en primer lugar a mi familia todo el apoyo que me ha brindado para acometer este gran proyecto. A mi mujer Verónica en especial, por todos los ratos y cosas que no hemos podido hacer en este tiempo; porque no he podido dedicar todo el tiempo que me haya podido necesitar para preparar la llegada de nuestro mayor regalo, nuestro hijo recién nacido Jorge. A Jorge, porque me ha dado el plus de motivación necesario para terminar este objetivo, y cualquier otro que me pueda proponer en adelante. A mis padres, por una vida dedicada a darnos lo mejor posible a mi hermano y a mí; fue emocionante ver la ilusión que les supuso conocer que me proponía a hacer la tesis. Y a mi hermano, al que también le he privado de tiempo para compartir ratos juntos, pero que seguro que lo compensaremos ahora con la llegada también de mi sobrino.

También quiero hacer un agradecimiento especial a mis directores de tesis, a Eladio Domínguez y a María Antonia Zapata, ya que son, sin duda, los responsables de que haya podido llegar hasta aquí. Desde el primer momento en una reunión, al principio de mi andadura de trabajo con ellos, cuando pensaron que tenía capacidad para afrontar este reto, y después por toda su dedicación para llevarlo adelante. A Eladio en particular también quiero agradecerle todo lo que me ha ayudado en mi desarrollo profesional; mucha parte de todo lo bueno que pueda conseguir viene de todo lo que he podido aprender en el máster y en Infozara.

A José Antonio Gutiérrez y a Roberto Barchino, artífices también de que hayamos podido concluir esta tesis. Muchas gracias por todo.

No me puedo olvidar de mis compañeros de Infozara y del grupo Noésis. A Antonio López, compañero de batallas en muchos proyectos y en el objetivo de la tesis en particular. A José Carlos Ciria (mil gracias por tus consejos y ayudas con ITIL, los indicadores, las bases de datos…), Beatriz, Raúl, María José, Chus, Ángel, Inés… todos siempre dispuestos a ayudarme en todo lo que he necesitado.

Tampoco puedo olvidarme de mis grandes amigos, en especial de Rubén, que ha sido mi gran motivador y consejero para que no decayera en ningún momento de este largo camino.

Resumen

Cualquier empresa que quiera competir en el mercado actual debe procurar la mejora continua de los procesos de negocio. Con este objetivo, en esta tesis se definen modelos y estructuras que permiten ayudar a la mejora de los procesos en dos vertientes: (1) mejorando los sistemas de monitorización existentes para la optimización de los procesos de negocio y (2) mejorando la información que se utiliza en los métodos de asignación de tareas para la ejecución de las tareas de los procesos por los recursos más adecuados. Para lograr estos propósitos, en la tesis se realiza una evolución de la consideración clásica de las bases de datos, en la que el dato es la unidad mínima de información, para definir una estructura que almacena como unidad mínima el conocimiento asociado a un hecho ocurrido en los procesos de negocio, el *acontecimiento*.

El concepto de acontecimiento posibilita almacenar en una misma estructura información que se necesita conocer sobre un hecho que se produzca en un sistema: qué ha ocurrido, quién lo ha realizado y cuándo se ha producido. Para ello, el acontecimiento se define como una estructura identificable e indivisible de acuerdo a tres dimensiones: *guía* –los aconteceres que ocurren-, *estructura* –los objetos de los aconteceres- y *comportamiento* –el efecto de los aconteceres, los cambios de estado o de la información asociada de los objetos-. La explotación de este conocimiento, almacenado en las bases de acontecimientos, nos permite obtener la *historia* de cualquier objeto del sistema, obteniendo la *línea de vida* de todos los acontecimientos que le han ocurrido en el transcurso del tiempo.

En el desarrollo de la tesis se define (1) el concepto de *acontecimiento* como unidad mínima de información de las *bases de acontecimientos*, las estructuras de persistencia de los *sistemas gestores de bases de acontecimientos* –SGBA-; (2) una *estrategia de diseño* para la construcción de sistemas de monitorización como SGBA, mediante la aplicación del concepto de acontecimiento al contexto de los procesos de negocio: el *acontecimiento basado en protocolos*; y (3) un patrón para la evaluación del desempeño de los recursos en base a la historia de realización de las tareas, el *Patrón de Acontecimientos basados en Tareas*, aplicando el concepto de acontecimiento al contexto de la gestión de tareas: el *acontecimiento basado en tareas*.

En el acontecimiento basado en protocolos, la estructura del acontecimiento se diseña para focalizar la explotación del conocimiento en poder conocer qué le ocurre a los objetos de un sistema en la ejecución de los protocolos que componen los procesos de negocio, sus cambios de estado y de información. La estrategia de diseño proporciona (1) una *metodología* para construir bases de acontecimientos basados en protocolos, definiendo patrones de diseño que permiten modelizar las bases de acontecimientos y su transformación a una plataforma de implementación específica, y (2) una *arquitectura* para la construcción de *Sistemas de Gestión de Bases de Acontecimientos basados en Protocolos* –SGBAP-, la cual permite construir sistemas de monitorización eficientes para realizar el análisis y mejora de los procesos de negocio en base al conocimiento almacenado en los acontecimientos.

En el acontecimiento basado en tareas, en cambio, la estructura del acontecimiento se diseña para conocer qué tareas han realizado los objetos del sistema, los recursos de la empresa, almacenando cuándo y con qué resultados las han realizado. Con este cambio de enfoque, el *Patrón de Acontecimientos basado en Tareas* nos proporciona un patrón de diseño de estructuras que permiten evaluar el desempeño de los recursos en base a su historia de acontecimientos. Para ello, se realiza una especificación de indicadores de rendimiento que

posibilitan esta evaluación del desempeño de los recursos sobre el conocimiento almacenado en los acontecimientos. La información que proporcionan estos indicadores nos permitirá mejorar la información de entrada a los métodos de asignación de tareas, como los métodos de optimización por programación lineal, precisando y actualizando los datos con la historia del desempeño de los recursos en la realización de las tareas a ser asignadas.

ÍNDICE

I. INTRODUCCIÓN Y PRELIMINARES

Capítulo 1. Introducción y objetivos

1.1 Introducción

Hoy en día las empresas compiten en un mercado globalizado y cada vez más competitivo que les exige una continua mejora en sus métodos de trabajo y en los resultados que producen, como productos y servicios.

Para ello, un elemento clave es la optimización de sus procesos de negocio, es decir, el conjunto de tareas y actividades relacionadas entre sí que se realizan para la obtención de un resultado –producto, servicio- [78].

En esta optimización de los procesos podemos identificar dos principales líneas de actuación:

- Por un lado, la optimización debe centrarse en la mejora del propio proceso de negocio, modificando y mejorando las propias tareas y sus relaciones para ser más eficientes en su ejecución.
- Por otro lado, la optimización también debe centrarse en la correcta asignación de las tareas a los recursos de la empresa, ya que una óptima ejecución de un proceso de negocio no sólo depende de que éste esté bien diseñado, sino que también depende de que sea realizado por los recursos adecuados.

Dentro de la primera línea de actuación, la gestión de procesos de negocio, conocida como *Business Process Management* -BPM- en inglés, es una disciplina dentro de la industria cuyo propósito es optimizar la realización de los procesos de negocio que se desarrollan en una organización [50][94].

Esta disciplina identifica las fases que componen el ciclo de vida de un proceso de negocio (diseño, modelización, ejecución, monitorización y optimización), y desarrolla metodologías, lenguajes y sistemas que permiten gestionar los procesos de negocio en base a dichas etapas, las cuales se repiten de forma continua durante la vida del proceso de negocio para conseguir cada vez mejores resultados.

Entre los sistemas que se implementan en el BPM, uno de los más importantes son los llamados sistemas de monitorización de procesos, conocidos como sistemas de *Business Activity Monitoring* –BAM- en inglés [23].

Estos sistemas son los encargados de registrar toda la información que se produce en la ejecución de los procesos, y son las herramientas principales que se utilizan en las etapas de monitorización y optimización, ya que permiten analizar e identificar los posibles errores y mejoras mediante el estudio de toda la información almacenada.

Sin embargo, estos sistemas suelen tener como problema que el registro de la información no se realiza de forma que su explotación sea eficiente, siendo en ocasiones meros sistemas de

log o registros en bases de datos sencillas que no permiten almacenar toda la información necesaria para un análisis completo y exhaustivo [36].

La segunda línea de actuación en la optimización de procesos de negocio, como hemos comentado anteriormente, trata de la optimización en la asignación de los recursos que realizan las tareas que componen los procesos, con el objetivo de reducir y optimizar tanto el tiempo como los costes de ejecución.

En la literatura, el concepto de asignación de tareas se utiliza en diferentes contextos y líneas de investigación en los que se requiere el estudio de cómo realizar una asignación eficiente. Una de las principales líneas de investigación se enmarca dentro de la investigación operativa [42], donde se aborda la resolución de los llamados problemas de asignación [17]. El problema de asignación es uno de los problemas de optimización matemática, y consiste en resolver cómo asignar de la manera más óptima los recursos que se disponen para la realización de un conjunto de tareas.

Esta tesis se centra en la investigación de soluciones y mecanismos que permitan mejorar ambas líneas de actuación en la optimización de procesos de negocio. Para ello, se desarrollan modelos que permiten, por un lado, mejorar los sistemas de monitorización de procesos para una mejor optimización de éstos y, por otro lado, mejorar la información que se utiliza en los métodos de asignación de los recursos que realizan sus tareas, permitiendo evaluar la historia del desempeño de los recursos.

1.2 Motivación y objetivos

El objetivo de esta tesis es desarrollar modelos y estructuras de persistencia que evolucionen la consideración clásica de las bases de datos, donde el dato es la unidad mínima de información, con el propósito de almacenar como un todo, como unidad mínima, todos los datos que son necesarios para conocer e identificar lo ocurrido en un hecho que se produce en un sistema de información: qué ha ocurrido, quién lo ha realizado y cuándo se ha producido.

Esta unidad mínima de información la llamaremos *Acontecimiento* y, al igual que en las bases de datos, definiremos el almacén que permite su persistencia en el tiempo, la *Base de Acontecimientos*, y el sistema que gestiona la información en estos almacenes de acontecimientos, el *Sistema Gestor de Bases de Acontecimientos.*

El desarrollo de estos modelos y estructuras nos posibilitará dar soluciones eficientes en la gestión de la información generada en la ejecución de procesos de negocio, en dos enfoques:

- Construir sistemas que permitan monitorizar y analizar de forma eficiente la evolución de la información que se dispone de los objetos de un sistema de información debido a la ejecución de los procesos de negocio; conociendo qué le ha ocurrido en cada momento, qué le ha producido y quién es el responsable de esos cambios.
- Construir sistemas que permitan evaluar el rendimiento de los recursos de la empresa -unidades funcionales- que ejecutan los procesos de negocio, con el objetivo de mejorar la asignación de tareas, definiendo indicadores de rendimiento sobre el conocimiento almacenado de todas las actividades que realizan en el desempeño de su trabajo.

Para lograr estos objetivos se define, por un lado, una estrategia de diseño para la

construcción de sistemas de monitorización de procesos, estableciendo: (1) una metodología mediante patrones y modelos de diseño para la construcción de las bases de acontecimientos, y (2) una arquitectura para la construcción de un sistema gestor que permita explotar la información almacenada en los acontecimientos, incluyendo tecnologías del Business Intelligence como el análisis de datos mediante cubos OLAP. Como prueba de concepto se describe la implementación realizada en un sistema para la gestión de muestras biológicas de un biobanco, llamado ARASIS.

Por otro lado, se define un patrón para la evaluación del rendimiento de las unidades funcionales utilizando el acontecimiento como estructura de persistencia, a través del cual se pueden establecer indicadores de rendimiento que permiten mejorar la asignación de tareas en base a la historia de su desempeño en el tiempo. En este caso, como pruebas de concepto se presenta la aplicación del patrón para la evaluación del desempeño en tres escenarios para demostrar su aplicación en diferentes contextos.

De esta forma, las aportaciones que realiza esta tesis son:

- Modificar la consideración clásica de las bases de datos definiendo una estructura mínima de información que permite almacenar, como un todo, toda la información requerida para identificar un acontecimiento en un sistema: qué ha sucedido, cuándo, quién lo ha realizado y cuál ha sido su resultado.
- Diseñar estructuras, modelos y sistemas que permiten obtener, gestionar y evaluar la historia de los objetos de un sistema de forma eficiente, posibilitando el desarrollo de análisis, métricas e indicadores sobre el conocimiento almacenado en los acontecimientos.
- Una estrategia de diseño para construir sistemas de monitorización de procesos de negocio eficientes que exploten el conocimiento almacenado en las bases de acontecimientos.
- Un patrón para la evaluación del desempeño de los recursos en base a la historia de realización de tareas, definiendo una especificación de indicadores de rendimiento que permitirán mejorar la información de entrada a los métodos de asignación de tareas.
- Proporcionar mecanismos para diseñar estos sistemas de forma independiente a las plataformas y tecnologías de implementación, suministrando los métodos y técnicas para transformar los modelos en plataformas concretas.

1.3 Estructura de la tesis

La tesis está estructurada en una serie de bloques en los que se desarrollan los objetivos de investigación:

1 En el bloque I se presentan los objetivos de investigación de la tesis -Capítulo 1- y los conceptos en los que se enmarca su desarrollo -Capítulo 2-, realizando una revisión de conceptos como la Gestión de Procesos de Negocio, los datos longitudinales o la asignación de tareas.

2 En el bloque II se realiza la definición de la base teórica en la que se sustenta el desarrollo de la tesis.

En el Capítulo 3 se definen de forma genérica los tres conceptos como evolución de la consideración clásica de las bases de datos:

- El concepto de Acontecimiento como unidad mínima de información.
- El concepto de Base de Acontecimientos como la estructura de persistencia de acontecimientos.
- El concepto de Sistema Gestor de Bases de Acontecimientos, como el sistema que permite gestionar y explotar el conocimiento almacenado en las Bases de Acontecimientos.

3 En el bloque III se aplican los conceptos anteriores elaborando una estrategia para desarrollar sistemas que permitan gestionar la información generada en los procesos de negocio que se realizan en una organización. Estos sistemas los llamaremos Sistemas de Gestión de Bases de Acontecimientos basados en Protocolos –SGBAP-.

En el Capítulo 4 se desarrolla el concepto de acontecimiento basado en protocolos, como aplicación del concepto de acontecimiento al contexto de los procesos de negocio, que permitirá definir estructuras para solventar las deficiencias existentes en los sistemas de monitorización actuales.

Sobre la base de este concepto, en el Capítulo 5 se define la estrategia de diseño, la cual se compone de tres elementos:

- Definición de patrones de diseño que permiten modelizar el concepto de acontecimiento basado en protocolos.
- Definición de una metodología para la construcción de Bases de Acontecimientos basados en Protocolos utilizando los patrones de diseño anteriores.
- Definición de una arquitectura para la construcción de Sistemas de Gestión de Bases de Acontecimientos basados en Protocolos.

Una versión inicial de la estrategia de diseño fue publicada en el artículo [27], publicado en la revista *IEEE Transactions on Knowledge and Data Engineering* indexada en el JCR.

En el Capítulo 6 se aborda uno de los problemas más complejos en los sistemas de monitorización, como es la adaptación a los cambios que pueden sufrir los procesos de negocio. Con este objetivo, se desarrollan una serie de mecanismos para abordar la evolución del sistema ante estos cambios.

Como prueba de concepto de la estrategia de diseño se presenta, en el Capítulo 7, la implementación realizada en ARASIS, una herramienta para la gestión de la información de un biobanco de muestras biológicas.

Además de ARASIS, durante el desarrollo de la estrategia de diseño, también se utilizan ejemplos de la aplicación de la estrategia al contexto de la gestión de incidentes según las buenas prácticas ITIL v3.

Por último, en el Capítulo 8 se presenta una estrategia alternativa en la metodología de construcción de las Bases de Acontecimientos basados en Protocolos, la cual es parte de la línea de trabajo futuro de la tesis. Esta alternativa tiene como objetivo optimizar la estrategia de diseño para obtener las líneas de vida de los objetos de forma más eficiente.

4 El bloque IV de la tesis consiste en la utilización del concepto de acontecimientos en la asignación de tareas para la evaluación del desempeño de las unidades funcionales que las realizan.

Este bloque se compone de tres elementos:

- Aplicación del concepto de Acontecimiento a la gestión de tareas, definiendo para ello el concepto de Acontecimiento basado en Tareas.
- Construcción de un patrón de diseño para la evaluación del rendimiento de las unidades funcionales, el Patrón de Acontecimientos basados en Tareas. La descripción de este patrón se publicó en el artículo [3], que fue presentado para el congreso EuroPloP del año 2016.
- Definición de indicadores de rendimiento sobre la historia almacenada en los acontecimientos que permitirán mejorar la información de entrada a los métodos de asignación de tareas.

5 En el bloque V se presentan las conclusiones obtenidas en la realización de la tesis, así como las líneas de trabajo futuras que se abren para continuar la investigación.

6 Por último, en el bloque VI se incluyen apéndices donde se presentan en detalle implementaciones realizadas en la prueba de concepto de ARASIS, como explicaciones de entornos desarrollados o la aplicación de la tecnología OLAP.

Capítulo 2. Marco de referencia

En este capítulo se presentan y describen los conceptos que han sido la referencia sobre los que se ha desarrollado la tesis.

2.1 Procedencia de los datos

El concepto de procedencia de un dato –*Data provenance* en inglés- refiere a la capacidad de conocer el origen de un dato, también llamado *linaje* del dato. El conocimiento concreto que se asocia con el *provenance* de un dato suele depender del contexto en el que se aplica [80][20]. Por ejemplo, en el contexto específico de las bases de datos, la procedencia de un dato se define como la descripción de su origen y los procesos que se han utilizado para su gestión. En el contexto de sistemas aplicados a entornos de negocio, la procedencia de un dato se refiere tanto al conocimiento sobre el origen de un dato registrado en el sistema (cuándo ha sido registrado, por quién, por qué…), así como todos los posibles cambios que haya sufrido a lo largo de su vida.

Actualmente existen muchas líneas de investigación en torno a posibles técnicas para gestionar la procedencia de los datos. Esto es debido en parte a la creciente necesidad que surge en Internet de asegurar esta información en los datos compartidos y enviados a través de la red; especialmente con la aparición de la Nube en Internet y todos los problemas de seguridad que conlleva respecto a los datos y su confiabilidad [7]. Debido al interés creciente por este asunto, el W3C publicó en 2013 un estándar, llamado PROV [90], estableciendo una especificación de un modelo de datos para intercambiar la procedencia de datos en Internet.

2.2 Gestión de Procesos de Negocio

La Gestión de Procesos de Negocio (Business Process Managament –BPM- en inglés) es una metodología empresarial cuyo objetivo es la mejora y optimización de los procesos de negocio de una organización, definiendo para ello métodos y técnicas que permiten modelar, diseñar, analizar y realizar el seguimiento de la ejecución de los procesos de negocio para su optimización [78][94].

El conjunto de herramientas de software que permiten realizar todas estas actividades sobre los procesos de negocio se denominan BPMS –Business Process Management Suite-, y entre ellas se encuentran herramientas para la modelización de los flujos de trabajo –workflows en inglés-, sistemas ERP de integración con otros sistemas de la entidad, y las llamadas

herramientas BAM –Business Monitoring Activity-, que son las herramientas encargadas de realizar el registro y monitorización de la información generada por los procesos de negocio [23].

Las herramientas BAM son las principales herramientas que se utilizan en las fases de monitorización y optimización de los procesos de negocio, ya que permiten consultar la información generada por los procesos de negocio y detectar posibles errores o malfuncionamientos. Sin embargo, como se ha comentado en la introducción de la tesis, estos sistemas normalmente almacenan la información en ficheros log o en estructuras que conllevan procesos complejos de obtención de los datos. Además, estas herramientas suelen tener el problema de que no almacenan toda la información necesaria para realizar un análisis completo de los procesos de negocio [74][14][36].

2.3 Dato longitudinal

Relacionado con el *data provenance* y la monitorización de procesos de negocio, se encuentra el concepto de dato longitudinal. Un dato, o conjunto de datos, puede considerarse longitudinal si refiere al mismo tipo de información de los mismos objetos o entidades y es resultado de realizar varias observaciones en distintos momentos a lo largo del tiempo [66][56]. Por ejemplo, la calificación media de un alumno en cada uno de los cursos de una carrera universitaria es un dato longitudinal del alumno. Otro ejemplo, en el contexto de los datos médicos de un paciente, sería los resultados de las analíticas de sangre a lo largo del tiempo, con objeto de conocer la evolución de una serie de valores determinados.

Cuando en el *data provenance* se habla del registro de los cambios que ha sufrido un dato, o cuando en los procesos de negocio se registra la información de un proceso a lo largo del tiempo, indirectamente estamos hablando de datos longitudinales.

El análisis sobre un dato longitudinal –*análisis longitudinal* o *estudio longitudinal*- nos permite conocer la evolución de un dato en el tiempo. A través de este análisis se pueden detectar, por ejemplo, tendencias en el tiempo, o posibles causas de las incidencias y sus consecuencias, es decir, aprovechar todos los beneficios que otorga la disponibilidad de una 'foto' del dato en diferentes momentos de su historia [57].

En función de cómo sean los intervalos de tiempo en los que se realizan las mediciones de los datos, el dato longitudinal podrá ser balanceado o no balanceado. Será un *dato longitudinal balanceado*, o lineal, cuando los intervalos y las entidades sobre los que se realizan las mediciones son regulares. Por el contrario, será un *dato longitudinal no balanceado*, o no lineal, cuando estos intervalos no sean regulares, existiendo periodos diferentes de medición para el dato longitudinal, o cuando entre diferentes mediciones varían las entidades que forman parte del dato longitudinal. Por ejemplo, imaginemos la evolución en el tiempo de la nota de matemáticas de un conjunto de alumnos. El dato longitudinal no sería balanceado en el caso de que los periodos de medición fueran indistintamente de un año, de dos, de tres… o de que alguna medición no se realizara sobre todos los alumnos del conjunto.

Uno de los aspectos más complejos del tratamiento de datos longitudinales radica en cómo estructurarlos de modo que se puedan almacenar y gestionar eficientemente. Conceptualmente hablando, existen dos formas de estructurar un dato longitudinal [56]:

- Representación multivariada. En este tipo de representación, el dato se representa en forma de matriz, de modo que en una misma fila se representan las diferentes mediciones que se realizan para una entidad. El tiempo de las mediciones se representa en la propia identificación de la columna. Por ejemplo, usar una matriz de calificaciones, guardando en cada fila todas las calificaciones de un alumno a lo largo de los últimos cinco años, siendo cada combinación asignatura-año una columna de la matriz.
- Representación univariada. En la representación univariada, cada medición realizada se representa en una fila de datos de la matriz, siendo el tiempo una columna más de datos. Por ejemplo, guardando en cada fila la calificación de un alumno indicando el año al que corresponde.

La representación multivariada tiene más problemas cuando los datos longitudinales no son balanceados, y cuando hay variación en los periodos de medición o en las entidades sobre las que se realiza cada medición. En esto casos, la representación multivariada puede dificultar el análisis longitudinal de los datos.

La representación univariada, por el contrario, se adapta a las situaciones que puedan presentarse, manteniendo el mismo tipo de estructura para realizar análisis longitudinales. En todo momento se mantiene el número de columnas, por lo que no hay que adaptar el análisis en función de las mediciones realizadas.

Si analizamos las dos propuestas anteriores, se puede observar que la representación univariada encaja mejor con la filosofía de las bases de datos, lo que nos lleva al concepto de base de datos longitudinal.

2.4 Bases de datos longitudinales

Una *Base de Datos Longitudinal* es una base de datos estructurada en la que se almacenan datos medidos a lo largo del tiempo, es decir, datos longitudinales.

Si analizamos la estructura de un dato longitudinal, se pueden identificar tres dimensiones [21][22][13]: *Entidad*, *Atributos* y *Tiempo*. Es decir, en un dato longitudinal se representa una entidad, un conjunto de datos de la entidad y un conjunto de datos temporales relacionados con los datos de la entidad.

El Grupo Noésis de la Universidad de Zaragoza, en cuya línea de investigación se enmarca la realización de esta tesis, desarrolló en el proyecto LISBioBank (*TSI-020302-2008-8; Subprograma Avanza I+D 2008*) [52] una teoría similar sobre el concepto de dato longitudinal, aunque con ciertas diferencias notables.

En primer lugar, contemplaba el concepto de Objeto en lugar de Entidad, ya que además de dotar de más expresividad al sistema en el que se utilizaba, permite utilizar las relaciones que son propias del paradigma de la orientación a objeto: relaciones entre objetos, estados, herencia… De esta forma, el dato longitudinal se convierte en un dato más estructurado, un dato estructurado por los estados de un objeto y por la estructura de éste.

Por otra parte, definieron el concepto de *hecho* o *evidencia* para indicar que todo dato que se registre como dato longitudinal debe incluir información que indique qué lo ha generado, es decir, el procedimiento que se ha ejecutado para su registro y el instante temporal en el que

se ha registrado.

Así, según la investigación realizada en el proyecto LISBioBank, el registro de un dato longitudinal implica el registro:

- Del objeto al que corresponden los datos.
- De los datos del procedimiento que se ha ejecutado.
- De los datos que ha generado esta ejecución.
- Del segmento temporal de validez de esos datos.

Esta idea de almacenamiento de los datos longitudinales fue la semilla, el origen, del concepto *Acontecimiento* que se define en el trabajo realizado en esta tesis.

2.5 Análisis de datos - tecnología OLAP

El *Business Intelligence*, o Inteligencia de Negocio o Empresarial, es un conjunto de técnicas, herramientas, métodos… cuyo objetivo es el análisis de toda la información generada en una organización, entre la que se encuentra, por ejemplo, la información generada por los procesos de negocio, información longitudinal… [86][35]

Una de las principales técnicas utilizadas en el Business Intelligence, es el análisis de datos que se realiza mediante la tecnología OLAP (acrónimo en inglés de "procesamiento analítico en línea" -On-Line Analytical Processing-) [19].

La tecnología OLAP es una solución utilizada para consultar y analizar grandes volúmenes de datos de una forma rápida haciendo uso de unas estructuras multidimensionales llamadas 'Cubos OLAP'.

El cubo OLAP es un conjunto de dimensiones relacionadas mediante una tabla de hechos, sobre las que se calculan un conjunto de medidas, siendo estos conceptos:

- Una medida es una cantidad que se quiere medir o cuantificar, por ejemplo, el número de ventas realizadas sobre un producto o el número de horas realizadas en la gestión de un proyecto.
- Una dimensión es una característica o atributo (o un conjunto de ellos), a través de los cuales se pueden obtener unos cálculos categorizados. Por ejemplo, se podría querer obtener el número de horas realizadas de un proyecto por el sexo del investigador, o por el departamento en el que se desarrolla. El sexo y el departamento serían dimensiones.
- La tabla de hechos es la tabla principal del esquema dimensional con el que se construye un cubo OLAP, ya que contiene los valores utilizados para calcular las medidas de negocio y las relaciones con las distintas dimensiones que forman el cubo OLAP.

Los cubos OLAP pueden diseñarse siguiendo diferentes esquemas alrededor de la tabla de hechos, siendo los más importantes el esquema en *estrella* y el esquema en *copo de nieve*.

Un esquema en estrella es un diseño en el que todas las dimensiones del cubo están relacionadas directamente con la tabla de hechos. Las dimensiones tienen una clave primaria simple, mientras que en la tabla de hechos, la clave principal es el conjunto de las claves

primarias de las dimensiones. Un ejemplo de esquema en estrella puede verse en la siguiente imagen:

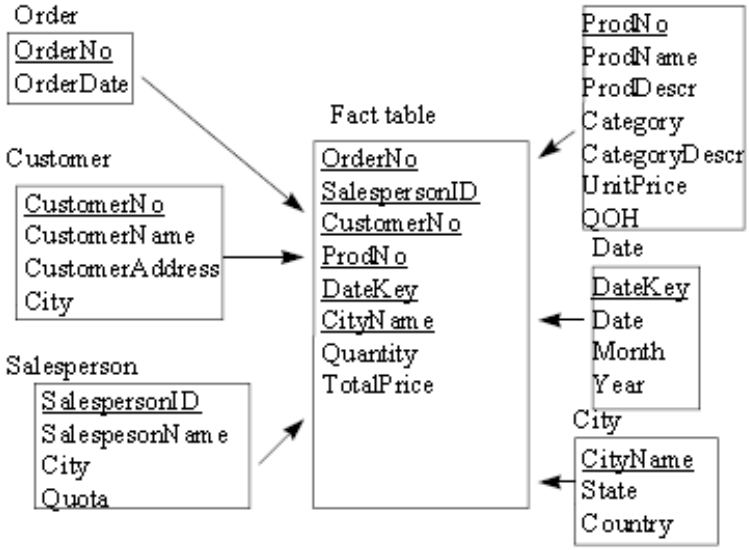


Figura 2-1 Esquema en estrella en [19]

Los esquemas en estrella son ideales por su simplicidad, proporcionando una gran eficiencia en análisis multidimensionales.

En los esquemas en copo de nieve las dimensiones pueden estar relacionadas indirectamente a la tabla de hechos a través de dimensiones intermedias:

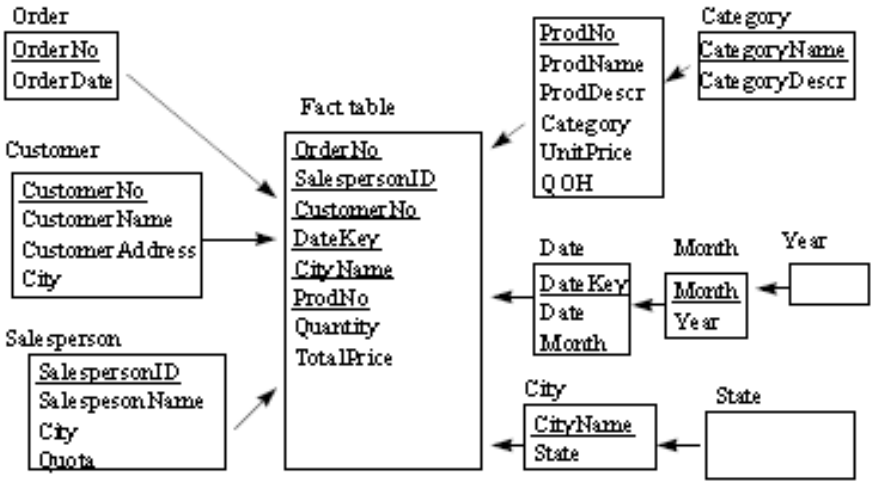


Figura 2-2 Esquema en copo de nieve o snowflake en [19].

La utilización de esquemas en copo de nieve evita la redundancia de datos gracias a la mayor normalización de las dimensiones que implica. Sin embargo, al añadir más tablas y relaciones entre ellas, se penaliza el tiempo de respuesta en la consulta de los datos.

Independientemente del tipo de esquema utilizado, en el cubo OLAP se podrían definir

medidas de sumas, medias, porcentajes… que nos permitan, por ejemplo, obtener la media de importe realizado en una venta por sexo del cliente o por la categoría de un producto. Las medidas definidas en un cubo OLAP constituyen otra dimensión interna del cubo (no tiene representación física en la tabla de hechos).

Si bien la tecnología OLAP está definida originalmente para trabajar sobre bases de datos multidimensionales, también se puede aplicar sobre bases de datos relacionales, existiendo tres tipos de sistemas OLAP:

- ROLAP: Consiste en la implementación de OLAP sobre una base de datos relacional. Para utilizar las bases de datos relacionales en un sistema OLAP, éstas tiene que estar convenientemente diseñadas, utilizando para ello, además de las propias tablas de las bases de datos relacionales, tablas o vistas adicionales (tablas resumen o agregaciones de datos confeccionadas expresamente para los cubos OLAP) que como resultado proporcionen una estructura de esquema multidimensional.
- MOLAP: Consiste en la implementación de OLAP sobre una base de datos multidimensional.
- HOLAP: Mezcla de las anteriores, en la que se almacenan algunos datos en una base de datos relacional, y otros sobre una base de datos multidimensional.

Una vez que se tiene definido el cubo OLAP, la información se consulta mediante el lenguaje MDX [60]. El lenguaje MDX es el lenguaje estándar de consulta a bases de datos multidimensionales sobre cubos OLAP. Una consulta MDX tiene una estructura muy similar a una consulta SQL:

```
SELECT
      {[Fecha].[Año].Members} ON COLUMNS,
      {[Measures].[Número ventas]} ON ROWS
FROM [CUBO_VENTAS]
WHERE {[Almacén].[Madrid Logística]}
```

Figura 2-3 Ejemplo consulta MDX

En la consulta anterior obtendríamos para cada año disponible, el número de ventas que se hayan producido en el almacén 'Madrid Logística'. La estructura de una consulta MDX se compone de la especificación de los miembros que se quieren mostrar en las columnas y en las filas de la tabla resultado, el cubo que se consulta y las condiciones que se quieran añadir a la consulta.

El lenguaje MDX se basa en los siguientes conceptos para definir los miembros que se incluyen en la zona de columnas, filas y condiciones:

- Dimensión: Representa una dimensión del cubo. Por ejemplo, la fecha de una venta o el almacén que la ha realizado.
- Nivel: Nivel dentro de una jerarquía de una dimensión. Una dimensión puede estar compuesta por una jerarquía de niveles. El ejemplo más habitual es el de la dimensión Tiempo, donde es usual que como niveles estén año, mes, día… organizadas en una jerarquía. Toda dimensión tiene al menos un nivel. Por ejemplo, el acceso al nivel Año sería [Tiempo].[Año]

- Miembro: Valor concreto de un nivel. Su notación es entre corchetes. El miembro 2015 del nivel Año en la dimensión Tiempo sería [Tiempo].[Año].[2015]. El miembro 2015 puede ser al mismo tiempo un nivel: [Tiempo].[Año].[2015].[Agosto] que representaría el miembro Agosto del año 2015.
- Tupla: Colección ordenada de uno o más miembros de una o más dimensiones diferentes. Su notación es entre paréntesis. Una tupla sería, por ejemplo, los productos y el año 2015: ([Tiempo].[Año].[2015], [Producto].[Nombre]). Los miembros de esta tupla serían todos los nombres de producto junto al año 2015: (2015,prod1), (2015,prod2), (2015,prod3)…
- Set –conjunto-: Colección ordenada de tuplas con la misma dimensión, su notación es entre llaves. Por ejemplo: {([Tiempo].[Año].[2015], [Producto].[Nombre]), ([Tiempo].[Año].[2016], [Producto].[Nombre])}. Los miembros de este conjunto serían todos los productos organizados por el año 2015 y el año 2016.

Al igual que ocurre con el lenguaje SQL, existen muchas funciones predefinidas para el lenguaje MDX referentes tanto a obtener valores numéricos ('Sum', 'Count', 'Avg'…) cómo funciones sobre los distintos conceptos anteriores (funciones para tratar dimensiones, miembros, tuplas…) que permiten filtrar, comprobar condiciones, obtener subconjuntos de miembros… Por ejemplo, la función 'Members' utilizada en el ejemplo de consulta MDX devuelve todos los miembros de una dimensión, un nivel o una jerarquía.

2.6 Asignación de tareas – Gestión de tareas

La Gestión de Tareas [59] es el proceso que se encarga de la gestión del ciclo de vida de una tarea, desde su preparación hasta su finalización. Este proceso, dada su estrecha relación con los procesos de negocio (los procesos son un conjunto de tareas relacionadas), se considera a veces como una parte del BPM. De hecho, la mayoría de herramientas software de BPM incluyen la gestión de tareas como una de sus funcionalidades.

Una de las etapas del ciclo de vida de la tarea es su asignación a un recurso disponible de la empresa para su realización. Esta actividad, que en un principio puede considerarse sencilla, puede suponer un problema de optimización importante cuando existen varias tareas a asignar y varios recursos que pueden realizarlas. Tanto es así, que la asignación de tareas es uno de los problemas de optimización definidos en la optimización combinatoria en matemáticas, denominado como *problema de asignación* [17].

El primer método conocido para resolver este problema fue el llamado Método Húngaro, llamado así debido a que este método fue inventado por matemáticos de esta nacionalidad. En concreto, en 1917 los matemáticos Dénes Köning y Jenö Egervary publicaron investigaciones para resolver este tipo de problemas de optimización, aunque no fue hasta 1955 cuando el también húngaro Harold W. Kuhn publicó el método húngaro, el cual fue revisado dos años más tarde por James Munkres en 1957 [65].

Actualmente, las técnicas de resolución más habituales de este problema son las técnicas de programación lineal y metaheurísticas, como parte de la rama de la investigación operativa [42]. En el campo de la robótica también se están realizando importantes avances en el ámbito de la asignación de tareas, ya que se está investigando en cómo asignar las tareas a

realizar por sistemas de robots de manera que puedan auto organizarse como si fueran sistemas inteligentes. En este ámbito, además de las técnicas habituales de programación lineal y metaheurísticas, se están realizando importantes desarrollos con las investigaciones en sistemas de inteligencias de enjambre, Swarm Intelligence en ingles [12], donde se estudia el comportamiento de sistemas distribuidos a modo y semejanza de sistemas naturales como las colonias de hormigas o abejas.

2.7 Proyecto AWHS: la herramienta ARASIS

La prueba de concepto de la estrategia de diseño que se desarrolla en el Capítulo 5 se ha realizado en el proyecto AWHS. El proyecto AWHS, 'Aragón Workers Health Study' [15], es un proyecto de investigación financiado por el CNIC, Centro Nacional de Investigación Cardiovascular, y el Gobierno de Aragón, cuyo objetivo general de investigación consiste en identificar aspectos genéticos y de hábitos vitales relacionados con factores de riesgo cardiovascular. Para conseguir este objetivo, se estudian las condiciones de salud durante un periodo de tiempo de un amplio grupo de trabajadores, cohorte de investigación, de la planta de construcción de automóviles de General Motors en Figueruelas, Zaragoza. El proyecto comenzó en 2009 y como resultado del proyecto se espera que, al final del estudio, el proyecto AWHS pueda proporcionar información novedosa sobre la distribución y trayectorias de los factores de riesgo.

Centrándonos en los aspectos relacionados con la tesis, uno de los objetivos del proyecto AWHS es el desarrollo de un biobanco de muestras biológicas de los trabajadores de la General Motors, tales como sangre, plasma o ADN. Estas muestras biológicas se procesan en varias alícuotas (se 'alicuotan', es decir, la muestra se divide en varias partes y se depositan en unos recipientes llamados alícuotas), las cuáles son almacenadas en contenedores adecuados, como cajas, racks o congeladores de muestras biológicas en el biobanco. Entre los tratamientos se encuentra tanto el procesamiento de las muestras para el estudio, como la cesión de muestras a otros laboratorios o su envío entre las diferentes sedes del biobanco (en el proyecto existe una sede en Madrid y otra en Zaragoza en las que se replican las muestras almacenadas por seguridad).

ARASIS es la herramienta informática que se ha desarrollado para registrar y gestionar toda la información de las muestras que se gestionan en el biobanco, permitiendo realizar un exhaustivo análisis de la información y dotando de los mecanismos necesarios para los estudios que se efectúan en el proyecto AWHS.

II. CONCEPTO DE ACONTECIMIENTO

Capítulo 3. Definición general del concepto de acontecimiento

En este capítulo se presentan los conceptos que conforman la base de conocimiento de esta tesis: Acontecimiento, Base de Acontecimientos y Sistema Gestor de Bases de Acontecimientos.

Estos conceptos suponen una evolución de la consideración clásica en las bases de datos, donde el dato es la unidad mínima de información, la base de datos la estructura de persistencia para gestionarlo, y el sistema gestor de bases de datos el motor necesario para explotar los datos almacenados en la base de datos. De forma análoga a esta consideración clásica, en primer lugar se define el acontecimiento como la unidad mínima de información que será almacenada en las bases de acontecimientos, las cuáles a su vez serán explotadas por los sistemas gestores de bases de acontecimientos.

En los siguientes apartados, estos conceptos se definen de forma abstracta, desarrollando qué se entiende por acontecimiento, por base de acontecimientos y por sistema gestor de acontecimientos. La aplicación de estos conceptos a diferentes contextos se realiza en las partes III y IV de la tesis, donde se diseñan estructuras de persistencia y arquitecturas para construir sistemas en base a ellos.

3.1 La noción de acontecimiento

En términos genéricos, un acontecimiento es algo que se percibe como el resultado de un acontecer. Por ejemplo: 'ha llovido', 'se ha obtenido una alícuota', 'se ha realizado un envío', 'se ha pagado una factura', 'Pedro ha construido una silla'.

En los ejemplos anteriores podemos percibir que hay un algo que ha concluido, que le ha acontecido a alguien o algo -ya sea una persona, una cosa, un lugar…-, o ha sido realizado por él, y con un resultado determinado. Dependiendo de la naturaleza del elemento y la acción del acontecimiento, estos acontecimientos los podríamos clasificar en dos tipos:

- Acontecimientos naturales.
- Acontecimientos artificiales.

Por acontecimiento natural se entiende cualquier suceso, acontecimiento, que se puede percibir en la naturaleza, como el ejemplo de 'ha llovido', o 'ha nevado en la sierra'. La particularidad de este tipo de acontecimientos es que el algo/alguien del acontecimiento puede no estar, o puede no ser, delimitado de forma concreta. En el ejemplo de 'ha llovido en la sierra' tenemos un claro exponente, 'la sierra' no es elemento u objeto que se puede

identificar de forma concreta, como una mesa o una persona. Además, la acción que acontece, el llover, es un fenómeno natural en el cual no se pueden establecer unos pasos o estructura para registrar cómo ha ocurrido.

El acontecimiento artificial, por su parte, recoge aquellos acontecimientos en los que sí se puede identificar el elemento de forma concreta, y la acción se puede estructurar en una serie de pasos o actividades, ya que son acontecimientos que surgen por la acción de sistemas construidos por el hombre, acciones que realizan las personas sobre dichos elementos, etc. En el ejemplo 'se ha obtenido una alícuota' se pueden identificar tanto el elemento, la alícuota, como un proceso estructurado en una serie de pasos, el alicuotado de una muestra biológica.

En esta tesis, nos vamos a centrar en los acontecimientos artificiales y, particularmente, en aquellos acontecimientos que suceden en entornos gestionados por sistemas de información. Si nos centramos, por tanto, en el contexto de los sistemas de información, ese alguien será un objeto del sistema sobre el que la acción de ese acontecer, o la realización de ese acontecer por el objeto, le ha generado un resultado, como cambiarle su estado o los datos que existan sobre él. Por ejemplo, en el caso de 'se ha pagado una factura', en un sistema de gestión de facturas, 'factura' es el objeto, el acontecer es el pago, y el resultado es que la factura queda en el estado 'pagada'. Sin embargo, el ejemplo de 'Pedro ha construido una silla' lo podemos interpretar en dos sentidos:

- La silla es el objeto sobre el que ha acontecido la acción de construir, quedando en estado construida. Asociada a la realización de la acción se podrá guardar que el ejecutor de la acción ha sido Pedro. Esta interpretación, en la que el objetivo es identificar qué ha ocurrido y sobre qué o quién, es lo que denominaremos acontecimiento basado en protocolos que se desarrolla en la parte III de la tesis.
- Pedro es el objeto, que realiza la acción de construir y cuyo resultado es la silla. En este caso, se podrá guardar que Pedro, como objeto, al realizar la acción cambiará su estado para reflejar la realización de la actividad. Este enfoque, por el contrario, en el que el objetivo es identificar quién o qué ha realizado la acción y qué acción ha realizado, es lo que denominaremos acontecimiento basado en tareas que se desarrolla en la parte IV de la tesis.

Por tanto, podemos concretar la definición general de acontecimiento con la que hemos partido, definiendo el acontecimiento como 'algo que ha acontecido sobre o por acción de un objeto y del que se deriva un resultado'.

Si analizamos la definición anterior, podemos identificar tres dimensiones: objetos, aconteceres y resultado. Los objetos en los sistemas de información constituyen el núcleo principal de la información que se almacena en ellos, su estructura, ya que establecen el alcance de la información que puede registrarse. Por ejemplo, en un determinado sistema de información se van a almacenar datos acerca de facturas, artículos, ventas… -objetos-. Los objetos a su vez se comportan en el sistema según sus diagramas de estado, donde se especifican sus posibles cambios de estado, o de sus datos, que pueden sufrir por la acción u ocurrencia de las condiciones que activan las transiciones de estado: los aconteceres. Es decir, los aconteceres guían el comportamiento de los objetos en el sistema.

De esta forma, podemos plantear una aproximación tridimensional en la que se considera simultáneamente la estructura de la información a almacenar (objetos), el comportamiento de estos objetos (estados, cambios de estado, cambios de datos) y las guías que les dirigen en sus comportamientos (aconteceres): el acontecimiento.

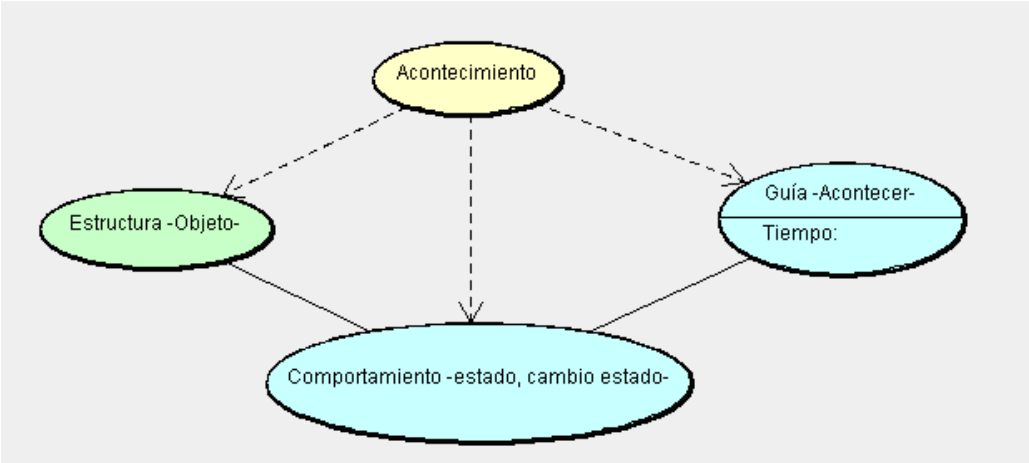


Figura 3-1 Concepto de acontecimiento

Mediante el concepto de acontecimiento logramos tener una estructura que contiene todos los elementos que necesitamos conocer de cualquier acción que se produce sobre o por parte de un objeto del sistema: qué objeto está implicado, qué ha sucedido y qué ha provocado. Además, añadiendo el atributo del tiempo en la guía, también podremos conocer cuándo ha sucedido el acontecer, información de especial importancia para completar la información del acontecimiento. Obtenemos, por tanto, un claro enriquecimiento del conocimiento que almacenamos al considerar en el acontecimiento estas dimensiones como un todo, y no como datos independientes en una base de datos convencional.

Por otro lado, podemos establecer que el acontecimiento es la unidad mínima de información en esta aproximación debido a que:

- Es único por naturaleza, ya que entre los elementos que lo definen está un acontecer específico, que es único cada vez que ocurre en el tiempo, siendo por tanto identificable sin ambigüedades.
- Cuando se identifica un acontecimiento, cada uno de los elementos que lo componen no son significativos por sí mismos.

Por último, podemos definir de forma genérica una representación del acontecimiento en forma de tupla de tres elementos, las tres dimensiones del acontecimiento:

(a, o, e)

donde *a* sería la representación de la acción, el acontecer, que actúa sobre o es realizada por un objeto *o,* provocándole el efecto *e*. La componte *e* se refiere a cualquier aspecto que suceda en el acontecimiento, ya sea registro de información o de cualquier cambio en el objeto, y para que la notación sea genérica, no se establece ningún formato o estructura concreta fija, sino que dependerá de qué se necesite diseñar en el sistema en el que se utilice el concepto de acontecimiento.

Por ejemplo, el acontecimiento 'se ha pagado una factura' que utilizábamos anteriormente, si ocurriera en un sistema, realmente lo que se tendría es que la factura –que se podría identificar, el ID del objeto, como 'f1'- ha cambiado al estado 'pagada' –cuyo identificador

podría ser ‘estPagada1’-, por la acción del acontecer ‘pagar factura’ -identificado como ‘pagFact1’-, lo que nos daría la tupla:

(‘pagFact1’, ‘f1’, ‘estPagada1’)

Como se puede ver en el ejemplo, el acontecimiento es la unidad mínima de información, ya que es único porque la acción ‘pagFact1’ se ha producido una vez en un momento del tiempo y ya no va a volver a repetirse (otra acción de pagar una factura será otro acontecimiento distinto, y se podrá identificar, por ejemplo, como 'pagFact2'), y cada elemento no son significativos por si mismos dentro del sistema:

- El estado ‘estPagada1’ no significa nada sin la factura ‘f1’, ya que se necesita saber de qué instancia de objeto es el estado.
- El acontecer ‘pagFact1’ requiere de los otros dos elementos para saber qué factura concreta se ha pagado y en qué estado la ha dejado.
- Sobre la factura ‘f1’ se necesita saber en qué estado está, y qué ha provocado que esté en ese estado.

Por tanto, podemos definir el concepto de acontecimiento, reuniendo todo lo anterior, como:

> Un ***acontecimiento*** es una pieza de información concreta, identificable e indivisible que contiene aspectos organizados de acuerdo a tres dimensiones: guía, estructura y comportamiento.

3.2 Base de acontecimientos (BA)

Una vez definida la unidad mínima de información en el marco de esta tesis, el acontecimiento, el siguiente paso es definir la estructura de persistencia que va a permitir registrarla, almacenarla y explotarla posteriormente.

De forma análoga a como se definen las bases de datos habituales que almacenan un conjunto de datos, donde el dato es su unidad mínima de información, podemos definir el concepto Base de Acontecimientos:

> Una ***base de acontecimientos*** –BA- (OcBase en inglés en analogía con Database) es un conjunto estructurado de acontecimientos.

La base de acontecimientos almacenará el conjunto de acontecimientos que se produzcan en un sistema, al igual que ocurre con los datos en una base de datos, considerando como ya hemos mencionado el acontecimiento como la unidad mínima de información. En la base de acontecimientos se tendrá que almacenar, por lo tanto, los elementos de las tres dimensiones que definen el acontecimiento:

- Estructura: Los objetos, junto con la información que los definan y las relaciones que existan entre ellos.
- Guía: Los aconteceres que actúan sobre o que realizan los objetos.
- Comportamiento: Los estados de los objetos, los cambios de estado o de información de los objetos por efecto de los aconteceres.

Como norma general de una base de acontecimientos, los acontecimientos registrados durante el transcurso del tiempo no serán eliminados, a diferencia de la filosofía de las bases

de datos tradicionales en las que cualquier dato puede ser eliminado siempre que se respeten los enlaces definidos entre tablas (claves foráneas).

La motivación de este planteamiento es debido a la naturaleza de los sistemas a los que están dirigidos las bases de acontecimientos. Como se ha comentado en la introducción de la tesis, uno de los objetivos es, entre otros, la utilización de las bases de acontecimientos en los sistemas de monitorización de procesos, cuyo propósito es monitorizar toda la información que acontece en la ejecución de un sistema, ya sea información correcta o errónea.

Para poder lograr este propósito, es necesario que no se elimine ningún acontecimiento que se haya registrado, ya que este hecho provocaría una pérdida irremediable en la información disponible sobre el sistema, que impediría, por ejemplo, una correcta ejecución de importantes procesos como los de auditoría. Además, la persistencia de todos los acontecimientos generados en el transcurso del tiempo permitirá el desarrollo de controles de calidad que ayuden a determinar las causas de esos errores y, por lo tanto, mejorar los sistemas de información en los que se utilicen las bases de acontecimientos. Sin embargo, como es obvio, este procedimiento general hará que el volumen de información registrada en la base de acontecimientos crezca a lo largo del tiempo, aumentando poco a poco los recursos necesarios para explotar el conocimiento registrado en ella.

En estos tiempos, en los que los sistemas van teniendo más recursos y más potencia de procesamiento, la necesidad creciente de recursos es un problema cada vez menor. Sin embargo, en ciertas situaciones se puede necesitar la eliminación de ciertos acontecimientos que están registrados en el sistema, y ya no tienen ninguna utilidad en él, con el objetivo de mejorar la eficiencia del sistema. Por ejemplo, en ciertos sistemas de gestión es necesario mantener la información generada y almacenada durante un determinado número de años por imperativo legal o administrativo de los financiadores. Este es el caso, por ejemplo, de la gestión de proyectos financiados por ministerios o en sistemas de gestión de salud, para poder atender cualquier requerimiento de información durante ese margen de años para justificaciones o estudios realizados. Sin embargo, pasados esos años, la información almacenada puede que únicamente tenga interés histórico. En estos casos, esos acontecimientos que ya no tienen utilidad real en el sistema, podrían ser almacenados en una base de acontecimientos paralela que almacene únicamente la información histórica, siendo eliminados de la base de acontecimientos de gestión del sistema, aumentando así la eficiencia de su funcionamiento.

Por otro lado, existen situaciones en las que sí que podría ser necesario eliminar algún tipo de información en las bases de acontecimientos dentro de la información de actualidad del sistema. Por ejemplo para la corrección de errores evidentes o para ajustarse a las necesidades o peticiones de algún tipo de ley en concreto. Es el caso, por ejemplo, del cumplimiento de leyes como la LOPD, donde se puede solicitar la eliminación o la anonimización (proceso por el cual se elimina la información registrada de su propietario real haciendo imposible su identificación) de datos registrados en la base de datos, especialmente en aquellos casos donde el nivel de protección sea alto, como los datos de salud. En el caso de errores evidentes, por ejemplo un individuo con un peso de 5.000 kg., se podría modificar al valor correcto, si se conociese, o eliminar el dato quedando como vacío, si se desconociese dicho valor. De este modo se pueden evitar posibles futuros errores en la utilización de ese dato (imaginemos por ejemplo el estudio del peso medio de todos los individuos, desvirtuaría el valor medio calculado).

No obstante, el sistema en el que se utilice la base de acontecimientos, para una gestión de calidad, debería permitir estos cambios o eliminaciones siempre que se registre una incidencia asociada donde se guarde que se ha modificado o eliminado la información con el motivo que ha provocado dicha acción. Si bien esto no puede ponerse como requisito ya que no siempre es posible una gestión de incidencias en los sistemas.

3.3 Sistema gestor de acontecimientos (SGBA)

De la misma forma que para gestionar una base de datos convencional se necesita un Sistema Gestor de Base de Datos (Oracle, SQLServer, PostgreSQL…), para gestionar una Base de Acontecimientos se necesita el concepto de un sistema gestor de acontecimientos:

> Un ***Sistema Gestor de Base de Acontecimientos*** –SGBA- (OcSystem en inglés) es cualquier sistema que permita la gestión completa de acontecimientos, tanto en términos de su almacenamiento (mediante el uso de una Base de Acontecimientos) como en términos de las herramientas que faciliten la explotación del conocimiento relacionado.

El SGBA, por tanto, se define sobre una Base de Acontecimientos, aumentando su funcionalidad ya que permitirá obtener los acontecimientos asociados a cualquier objeto específico o a cualquier acontecer que le hubiera acaecido. Para lograrlo, un SGBA estará dotado de diferentes estructuras y mecanismos, tanto para la gestión de los acontecimientos como para su explotación. A través de un SGBA será necesario poder registrar los acontecimientos de los objetos que se vayan produciendo a lo largo del tiempo, así como poder buscar un acontecimiento concreto que se haya producido en un momento determinado.

Por otro lado, un SGBA deberá proporcionar mecanismos que permitan analizar y explotar el conocimiento almacenado en los acontecimientos, permitiendo establecer y obtener las relaciones que existan entre ellos. Entre estos mecanismos se puede destacar la funcionalidad necesaria para obtener estructuras tales como la línea de vida y la historia de un objeto.

> La ***historia*** de un objeto es el conjunto de todos los acontecimientos relacionados con ese objeto.

Hay que hacer notar que, con el término 'acontecimientos relacionados ' se hace referencia tanto a los acontecimientos ocurridos directamente en el objeto, como a los ocurridos en objetos que están relacionados con él. Por ejemplo, en el contexto de la compraventa de artículos, en la historia de un artículo de venta también hay que considerar los acontecimientos que le han sucedido a la caja en la que ha sido enviado, ya que forman parte, aunque de forma indirecta, de su historia.

Los acontecimientos que han ocurrido directamente en el objeto en sí, serán su línea de vida en el sistema.

> La ***línea de vida*** de un objeto es el conjunto de acontecimientos que han ocurrido en un objeto.

De hecho, la línea de vida de un objeto siempre es un subconjunto de los acontecimientos que componen su historia, siendo la historia un conjunto de líneas de vida de objetos relacionados.

A través del SGBA se podrá obtener, por tanto, una red de líneas de vida de objetos a través de las propias relaciones entre los objetos, lo que permitirá filtrar y buscar acontecimientos mediante condiciones de búsqueda generales. Por ejemplo, en el caso de la línea de vida de un artículo en concreto, ésta podrá ser filtrada para conocer la evolución de su precio a lo largo del tiempo, obteniendo en este caso de forma separada los instantes en los que ha variado de precio. En el caso de varias líneas de vida, un ejemplo sería obtener las líneas de vida de todos los artículos cuya caja de envío haya pasado por una ciudad en concreto.

De esta manera, la red de líneas de vida que proporciona un SGBA constituye un 'lenguaje de acceso', permitiendo obtener información derivada de los acontecimientos que han ocurrido y facilitando la monitorización del conocimiento.

Por último, conviene destacar que este tipo de gestión de la información hace que un SGBA sea un sistema longitudinal, puesto que es capaz de informar lo que le ha ocurrido a un objeto en un momento puntual del tiempo o durante un periodo.

III. ACONTECIMIENTOS BASADOS EN PROTOCOLOS

Capítulo 4. Sistemas de monitorización de procesos: Acontecimientos basados en protocolos

La gestión de procesos de negocio –Business Process Management, BPM- consiste en el estudio, análisis y modificación de los procesos de negocio con el objetivo de realizar su mejora continua [78][50]. El BPM es una disciplina en pleno auge desde hace varias décadas dentro de la industria, ya que permite a las empresas analizar su método de trabajo para conseguir cada vez mejores productos y/o resultados.

Una de las principales fases del BPM es la monitorización de los procesos de negocio -Business Process Monitoring-, que consiste en el seguimiento en tiempo real de la ejecución de dichos procesos. Para este seguimiento se hace uso de los sistemas de monitorización de procesos, que son los encargados de recoger la información que se genera durante su ejecución, con el objetivo de poder analizarla y explotarla en la consecuente modificación y mejora de ellos.

En este capítulo se presenta una estrategia de diseño para construir sistemas de monitorización de procesos utilizando el concepto de acontecimiento (en particular, la aplicación del concepto de acontecimiento a los procesos de negocio, el acontecimiento basado en protocolos), a los que llamaremos Sistemas de Gestión de Bases de Acontecimientos basados en Protocolos –SGBAP-. Para ello, primero se presentan los problemas que tienen actualmente los sistemas de monitorización para realizar una explotación eficiente de la información. Después se define de forma abstracta el concepto de acontecimiento basado en protocolos, como caso particular del concepto de acontecimiento. A continuación, se describe la estrategia de diseño que proporciona una metodología y una arquitectura para la construcción de SGBAPs. Como parte de la estrategia, también se muestra la escalabilidad de estos sistemas ante la aparición de nuevos procesos en el sistema –evolución ante el cambio-, y se presenta un caso de aplicación en el mundo real, el caso ARASIS (presentado en el apartado 2.7 de esta tesis), que evidencia su viabilidad.

Una versión inicial de la estrategia de diseño desarrollada en la tesis fue publicada en el artículo [27], publicado en la revista *IEEE Transactions on Knowledge and Data Engineering* que está indexada en el JCR. Respecto a la versión publicada, en el documento final de tesis se profundiza en los patrones de diseño desarrollados en la estrategia, rediseñándolos y ampliándolos, para contemplar no sólo los cambios de estado de los objetos –versión inicial- sino también los datos longitudinales que pueden generarse por efecto de la ejecución de los protocolos.

Por último, se presenta una alternativa a la estrategia de diseño anterior que propone un cambio en la forma de almacenar el efecto producido por los acontecimientos. La principal ventaja de la nueva estrategia es que se consigue mejorar la forma de obtener las líneas de vida de los objetos, haciendo más eficientes los sistemas SGBAPs.

Para explicar los conceptos y diseños presentados en los apartados siguientes se incluyen ejemplos utilizando el propio caso ARASIS y el proceso de gestión de incidencias de las buenas prácticas más reconocidas en la gestión de servicios de la información, la Biblioteca de Infraestructura de Tecnologías de Información (ITIL por sus siglas en inglés [10]).

4.1 Sistemas de monitorización de procesos

En la gestión de procesos de negocio se puede identificar un conjunto de fases que permiten diseñar y mejorar los procesos de forma cíclica, denominado el ciclo de vida BPM:

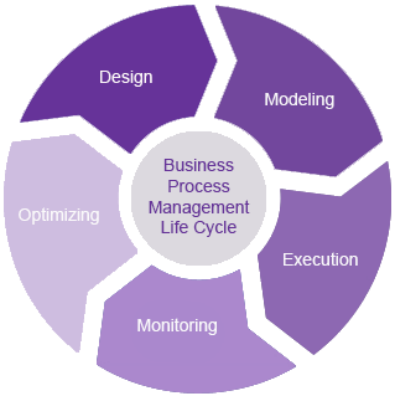


Figura 4-1 Ciclo de vida BPM. Imagen obtenida de http://leapforward.illinoisstate.edu/BPM

En la literatura existen diferentes versiones de las fases que componen el ciclo de vida BPM [78][94][48][87], aunque se pueden resumir principalmente en las cinco que se pueden ver en la imagen anterior:

- Diseño: En esta fase se identifican las actividades que componen los procesos, los actores, los flujos teóricos de trabajo de los procesos –workflows-, se aplican en el diseño las posibles mejoras identificadas en el proceso de optimización…
- Modelización: Una vez realizado el diseño del proceso de negocio, se construye el modelo con una notación específica –por ejemplo, con lenguaje BPMN [67]-, se simula su ejecución para comprobar su correcto funcionamiento… Para ello se hace uso de herramientas informáticas especializadas en la modelización de procesos de negocio.
- Ejecución: Con el modelo del proceso de negocio correctamente desarrollado, éste se aplica en el entorno real de negocio.
- Monitorización: Durante la ejecución de los procesos se registra la información que se genera, se obtienen informes de seguimiento, se registran posibles incidencias, se muestran avisos de cualquier posible mal funcionamiento…
- Optimización: La información obtenida durante la monitorización se analiza para identificar posibles problemas existentes, evaluarla, y poder así mejorar y optimizar los procesos de negocio. En muchas ocasiones, la fase de optimización se considera junto a la fase de monitorización como una única fase conjunta, ya que suele ser habitual que

se solapen en el tiempo.

Los sistemas de monitorización de procesos (llamados sistemas BAM en inglés –Business Activity Monitoring- [23]) son las herramientas utilizadas durante las fases de monitorización y optimización. Estas herramientas realizan el registro de la información de la ejecución de los procesos de negocio y se utilizan para analizar los procesos y llevar a cabo su mejora continua [48][14][47].

Esta información se registra habitualmente como una secuencia de eventos [76] que se producen en el sistema durante la ejecución del proceso, registrándose normalmente el evento, el momento en el que se produce, su ejecutor y algunos datos adicionales que sean necesarios según el contexto de los procesos [74][88]. Un ejemplo es el caso del envío de mercancías, donde se registran los puntos del recorrido con la indicación de la hora de llegada a cada punto, el transportista que ha realizado esa fase del trayecto y otras informaciones relacionadas.

El problema habitual que existe en este tipo de sistemas es que este registro se realiza usualmente mediante sistemas que impiden una eficiente explotación de dicha información, o que necesitan de la utilización y desarrollo de complejos sistemas de minería de datos -*data mining*- [74]. Por ejemplo, se suelen registrar los datos en ficheros de texto plano (*logs*), en estructuras lineales de bases de datos (como una tabla con la información distribuida en diferentes campos) o incluso en hojas de cálculo [74][14][36].

Por otro lado, también es frecuente que estos sistemas no recojan toda la información que se necesita para una correcta monitorización y análisis. Por ejemplo, no se suelen registrar los objetos que participan en el evento, sus posibles cambios de estado, o los protocolos seguidos por el usuario; información que puede ser importante en la mejora de los procesos de negocio como, por ejemplo, en los procesos de auditoría o para obtener correctamente el *provenance* de los datos [80].

Teniendo en cuenta los problemas anteriores, vamos a plantear una estructura más compleja que nos permita almacenar más información de cada evento que sucede en el sistema. El objetivo es facilitar su explotación y poder diseñar sistemas de monitorización que sean escalables y adaptables a los cambios que se produzcan en el negocio.

4.2 La noción de acontecimientos basados en protocolos

En la norma ISO 9000 -documento que registra las definiciones para los Sistemas de Gestión de la Calidad [43][44]- se define **proceso** como un "conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados". Por ejemplo, para la extracción de muestras biológicas, existe un proceso que establece las actividades que hay que llevar a cabo para obtener una muestra de un sujeto de estudio y almacenarla correctamente en congeladores. Otro ejemplo lo podemos encontrar en el proceso de la gestión de las incidencias de ITIL, en el que se establecen las actividades a desarrollar para resolver las incidencias que suceden en un servicio o sistema: registro, clasificación, análisis, resolución…

Cuando se especifica una forma concreta de cómo ejecutar un proceso, lo que estamos definiendo es, en notación de la ISO 9000, un **procedimiento** para llevarlo a cabo. Según la

norma ISO 9000, un procedimiento es "una forma específica para llevar a cabo una actividad o un proceso". En esta definición se puede observar que el procedimiento se puede definir tanto a nivel de proceso global como de una actividad concreta del proceso. Es decir, podemos definir un procedimiento que especifique cómo realizar todo el proceso en conjunto, o podemos definir un procedimiento concreto para cada actividad del mismo (o para un conjunto de ellas). En este último caso, el proceso estará compuesto por un conjunto de procedimientos.

A este concepto de procedimiento en esta tesis lo llamaremos '**protocolo**' en analogía con los protocolos médicos o los protocolos de calidad, los cuales establecen formas concretas de realizar los procesos médicos y de gestión de calidad en sistemas respectivamente. Por tanto, podemos definir los conceptos de proceso y protocolo en el marco de esta tesis:

> Un ***proceso*** es una tarea de principio a fin, identificable y concreta realizada para un propósito dado, conceptualizado como un conjunto cohesionado de protocolos.
>
> Un ***protocolo*** es una especificación concreta de un orden lógico y estructurado de actividades diseñadas para lograr el objetivo de un proceso.

Siguiendo con los ejemplos anteriores, para el proceso de obtener muestras biológicas, podremos tener definidos, por ejemplo, el protocolo de alicuotado de las muestras (que establece los pasos concretos que hay que seguir para dividir las muestras en alícuotas de forma correcta) o el protocolo de almacenamiento de las muestras en congeladores (que define cómo hay que almacenarlas para su correcta conservación en los biobancos). En el proceso de la gestión de las incidencias por su parte, podemos descender a un nivel inferior definiendo un protocolo para cada actividad del mismo: protocolo de registro de incidencia, protocolo de clasificación, protocolo de análisis, protocolo de resolución…

Es importante destacar que el protocolo establece una forma concreta de realizar un proceso o conjunto de actividades en un momento determinado, pero esto no quiere decir que sea la única forma, por lo que para un mismo proceso, los protocolos que están definidos para él pueden cambiar en el tiempo definiendo otras formas de implementarlo. D este modo podremos abordar el problema de adaptar los sistemas de monitorización de procesos ante cambios en el negocio, pudiendo gestionar las posibles variaciones en los protocolos de un proceso. Esto se explica más en detalle en el Capítulo 6.

4.2.1 Definición de acontecimiento basado en protocolos

Cuando se ejecuta un protocolo, éste actúa sobre uno o más objetos del sistema a los que cambiará su estado o la información que hay almacenada sobre ellos. Esta acción la podemos denominar ***ejecución de protocolo***, y es realizada por un actor o ***ejecutor de protocolo*** (una persona, una máquina, una aplicación…).

Los posibles cambios de estado de los objetos, o de los datos relacionados, son el ***comportamiento*** de dichos objetos en los procesos, siendo el cambio concreto el ***efecto*** que produce una ejecución de protocolo sobre un objeto.

Si recordamos el concepto general de acontecimiento lo hemos definido como "una pieza de información concreta, identificable e indivisible que contiene aspectos organizados de acuerdo a tres dimensiones: guía, estructura y comportamiento".

Tal como hemos descrito la ejecución de un protocolo podemos identificar las tres dimensiones de un acontecimiento: la estructura de la información a almacenar (objetos), el comportamiento de estos objetos (estados, cambios de estado, cambios de datos) y las guías que les dirigen en sus comportamientos (protocolos y ejecutores de protocolo). Por ejemplo, siguiendo el ejemplo anterior del protocolo de alicuotado de muestras, su ejecución (guía) causa, entre otras cosas, que una parte de la muestra –la alícuota- (estructura) sea almacenada en un contenedor –llamado 'rack'- (comportamiento).

De esta forma, podemos extender la definición de acontecimiento para definir el acontecimiento basado en protocolos:

> Un ***acontecimiento basado en protocolos*** es una pieza de información concreta, identificable e indivisible que contiene aspectos organizados de acuerdo a tres dimensiones -guía, estructura y comportamiento- proporcionados por una ejecución de protocolo, un objeto y un efecto.

Es importante destacar que, al igual que el acontecimiento general, el acontecimiento basado en protocolos sigue siendo la unidad mínima de información debido a que:

- Es único por naturaleza, ya que entre los elementos que lo definen está la ejecución del protocolo, que es única cada vez que ocurre en el tiempo, siendo por tanto identificable sin ambigüedades.
- Cuando se identifica un acontecimiento, cada uno de los elementos que lo componen no son significativos por sí mismos.

Por último, como se ha comentado al principio del apartado, una ejecución de protocolo puede actuar sobre varios objetos provocándoles un efecto como cambiar su estado. Sin embargo, para que sea unidad mínima de información, un acontecimiento refiere a un efecto concreto sobre un objeto concreto. Por este motivo, en ese supuesto lo que produce dicha ejecución de protocolo son varios acontecimientos basados en protocolo. Por ejemplo, siguiendo con el ejemplo del protocolo de alicuotado de muestras, en dicho protocolo se establece, como se ha comentado, el contenedor -'rack'- en el que son depositadas las alícuotas. En un determinado momento, este contenedor puede completarse al llegar a su capacidad máxima, cambiando éste de estado a 'completado', por lo que la ejecución de protocolo generaría dos acontecimientos diferentes: el que la alícuota esté almacenada y el que el rack esté completado.

4.2.2 Representación de un acontecimiento basado en protocolos

El concepto de acontecimiento, de forma general, hemos dicho que se puede representar mediante una tupla de tres elementos:

(a, o, e)

donde *a* es la representación de la acción, el acontecer, que actúa sobre un objeto *o,* provocándole el efecto *e*.

En el caso del acontecimiento basado en protocolos lo podemos representar, por tanto, por una tupla igual de tres elementos

(ep, o, d)

donde *ep* es la ejecución de protocolo, el acontecer, que actúa sobre un objeto *o,* dando lugar a los datos *d* asociados al cambio en el objeto en un momento determinado del tiempo (el efecto).

Como se ha comentado en la definición del acontecimiento basado en protocolos, la componente *d* se refiere a cualquier cambio que le suceda al objeto por acción de la ejecución del protocolo. Esta componente puede ser información asociada al objeto o cualquier cambio de estado en él, sin que tenga un formato o estructura fija para que esta notación sea genérica. Por ejemplo, en el contexto de la gestión de muestras biológicas, podría existir un protocolo de extracción de parte del contenido de una alícuota, en el cual se modifica la información de la cantidad de muestra recogida en ella, siendo un acontecimiento posible:

(ejecProtocoloExtraccionAlicuotado001, alicuota001, (50, ml))

En este acontecimiento, se dispone de la información de que el objeto 'alicuota001' se ha obtenido en la ejecución de protocolo 'ejecProtocoloExtraccionAlicuotado001' con los datos de '50 mililitros'. En este caso, en la componente *d* se almacena tanto la cantidad de la muestra -50- como la unidad de recogida –ml-.

Otro ejemplo de acontecimiento podría ser, esta vez en el contexto de la gestión de incidencias, la clasificación de un incidente mediante la ejecución del protocolo de clasificación de incidentes 'ejecProtocoloClasif001', cambiando el estado del incidente 'incidente001' de registrado (identificado como 'incRegistrado001') a clasificado (identificado como 'incClasificado002') mediante una transición de estados ('incRegistrado001CambiaAIncClasificado002'):

(ejecProtocoloClasif001, incidente001, incClasificado002)

Siendo 'incClasificado002', la componente *d*, el estado al que cambia el objeto incidente como efecto de la ejecución del protocolo a través de la transición.

4.2.3 Efecto de los acontecimientos: estados y datos

Como se establece en la definición de un acontecimiento basado en protocolos, en el efecto producido por una ejecución de protocolo sobre un objeto puede ocurrir que cambie el estado del objeto o la información de los datos que existen sobre él.

El estado de un objeto representa las características del objeto en un momento dado dentro del sistema. Por ejemplo, en el contexto de la gestión de incidencias según ITIL v3, un incidente en el sistema de gestión puede estar en el estado de registrado (el incidente se ha identificado y registrado en el sistema), clasificado (se ha realizado la clasificación del incidente determinando su prioridad, urgencia…), analizado (se han analizado las causas del incidente y sus posibles soluciones), etc., hasta el estado cerrado, que es cuando ha terminado la gestión del incidente.

En los diagramas de estado se representan conceptualmente estos estados del objeto, con las posibles transiciones de estado que existen y los eventos o acciones que provocan cada transición que, como hemos comentado en el acontecimiento basado en protocolos, son las ejecuciones de protocolo en nuestro contexto. En la siguiente imagen puede verse un ejemplo del diagrama de estados de un incidente:

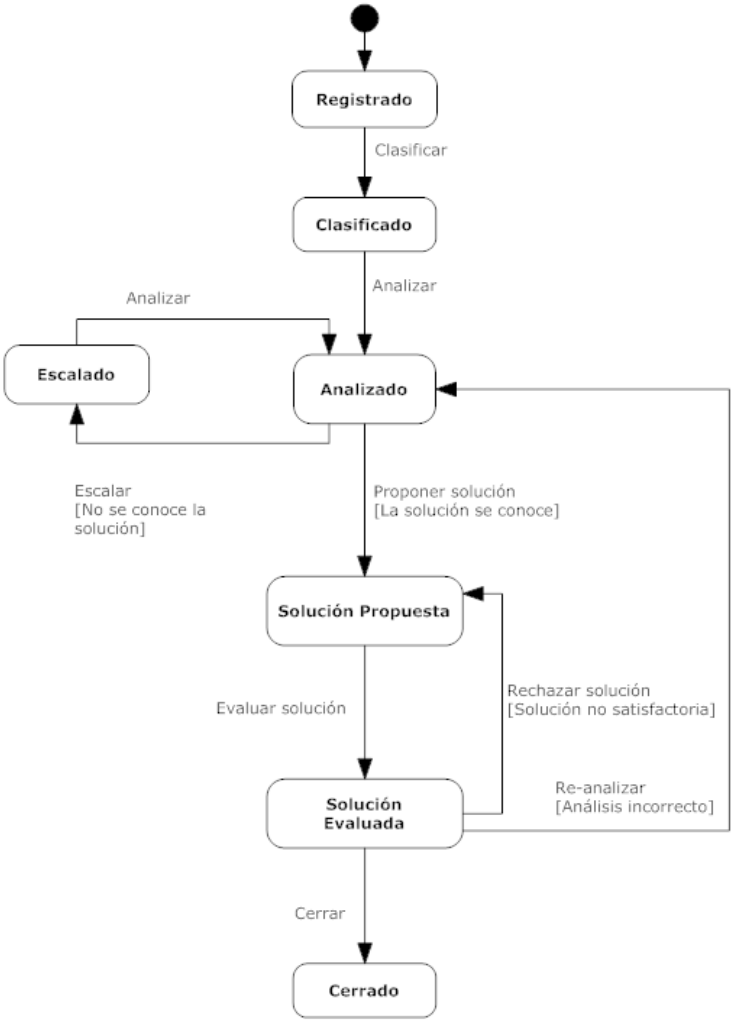


Figura 4-2 Diagrama de estados de un incidente

En general, estos diagramas de estado serán diagramas finitos de estados [95], existiendo un estado inicial y un estado final que marcarán los límites de la vida de un objeto en el sistema mediante un número finito de estados posibles, estableciendo el ciclo de vida del objeto. Por ejemplo, el incidente nace en el estado registrado y muere en el estado cerrado o, por considerar otro contexto diferente, en los sistemas de ventas un pedido nace cuando se realiza el pedido –estado realizado- y finaliza, por ejemplo, cuando ha sido enviado –estado enviado-.

Asociados a estos cambios de estado normalmente existirá información del objeto, datos, que serán almacenados en la base de acontecimientos cuando ocurran las correspondientes ejecuciones de protocolo. Estos datos podrán ser estáticos –invariantes en el tiempo- o dinámicos –datos dependientes del tiempo en el que fueron generados-. Las características que definen a un objeto y que se registran al crearlo son datos estáticos, como por ejemplo, el identificador de un incidente o el código de una factura. Los datos estáticos no varían durante el tiempo, o si lo hacen no interesa almacenar su evolución, y debido a ello suelen estar asociados directamente con el objeto como datos dependientes de éste.

En los datos dinámicos, por el contrario, sí que se necesita almacenar los distintos valores que han tenido en el tiempo. Pensemos por ejemplo en la gestión de un incidente del diagrama de estados anterior en la Figura 4-2. En el diagrama puede verse que el incidente puede ser re-analizado debido, por ejemplo, a que la solución propuesta de un análisis no fuera correcta, requiriendo ser analizado de nuevo para buscar una nueva solución. En este caso, si queremos preservar los distintos análisis que se realizaron en el incidente junto a la información que nos proporciona el acontecimiento basado en protocolos (qué los generó –el protocolo-, quién –el ejecutor- y cuándo), los datos deberán estar asociados al estado 'analizado' en el que fueron generados (por la ejecución del protocolo de analizar el incidente).

En los acontecimientos basados en protocolos, este tipo de datos anteriores, tanto estáticos como dinámicos, estarán dentro de la componente 'd' de la tupla que define el acontecimiento, como datos asociados al estado en el que fueron generados. Por ejemplo, el registro del incidente podríamos representarlo de forma genérica como:

(ejecProtocoloRegistr001, incidente001, (incRegistrado001, (descripción, fecha))

En este ejemplo, los datos estáticos del incidente (descripción y fecha en la que ha ocurrido) están asociados al estado 'incRegistrado001'.

Sin embargo, la ejecución de un protocolo también puede generar datos asociados a un objeto que no impliquen un cambio de estado de éste, sino un registro longitudinal de datos sobre el objeto. A este tipo de datos los llamaremos **datos longitudinales**, ya que en ellos se almacenan la evolución de un dato en el tiempo. Por ejemplo, pensemos en el registro de análisis clínicos sobre un objeto paciente, a través de un protocolo de registro de análisis. En este caso, el dato longitudinal -el análisis clínico- es un objeto más del sistema como tal, ya que tiene entidad propia, pero estará asociado al objeto del que es información longitudinal –el paciente-.

La diferencia del objeto del dato longitudinal, con respecto al objeto al que está asociado (análisis clínico y paciente en el ejemplo anterior), es que el dato longitudinal no tiene estados que definan su comportamiento en el sistema, sino datos concretos que lo definen (el análisis clínico se compone de las diferentes determinaciones que se analizan en él –glucosa, hemoglobina, hierro…-). La consideración del análisis como dato longitudinal lo otorga la posibilidad de poder ejecutar el protocolo de registro tantas veces como se requiera en el tiempo, registrándose por tanto tantos análisis como ejecuciones del protocolo existan. Cada ejecución tendrá asociado la información del tiempo en el que fue realizado el análisis. De esta forma, podremos obtener la evolución en el tiempo de sus datos. Por ejemplo, uno de los datos de los análisis clínicos será el nivel de glucosa en la sangre. Este dato estará registrado en cada análisis registrado en el tiempo, por lo que podremos conocer la evolución en el tiempo de la glucosa del paciente al que pertenezcan los análisis.

Aunque los datos longitudinales no tengan estados por sí mismos, si analizamos en abstracto la asociación del dato longitudinal con el objeto, podemos considerar la información de estos datos como unos estados especiales del objeto en cuestión. Siguiendo con el ejemplo de los análisis clínicos, sus datos nos proporciona la información del estado clínico del paciente, donde cada estado corresponde a cada análisis clínico que se ha registrado. Sin embargo, en este caso no existe un diagrama de estados finitos con la indicación de los protocolos que provocan el cambio de estado, sino que el cambio de estado se produce por el registro nuevo de un análisis, siendo el estado inicial el primer análisis registrado y el estado final el último

análisis.

De esta forma, en los datos longitudinales se tiene un diagrama infinito de estados, lo que se conoce en matemáticas como un infinito potencial [33], ya que el estado final del diagrama es desconocido (siempre puede existir un nuevo análisis), pero cada estado es finito y se alcanza en un número determinado de pasos (marcados por las ejecuciones del protocolo en el tiempo). La representación abstracta del dato longitudinal en nuestro contexto se puede ver en la siguiente imagen:

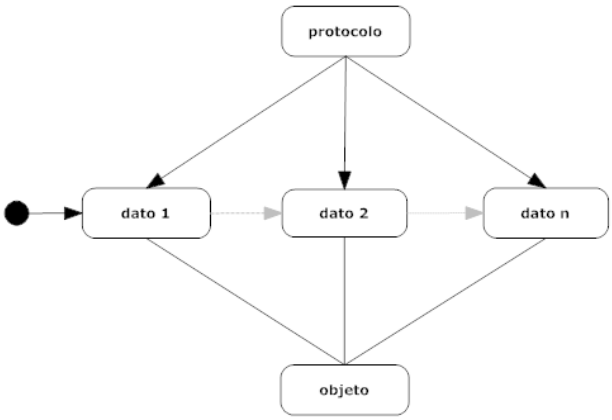


Figura 4-3 Representación de un dato longitudinal

Los datos, asociados al objeto, son generados por las ejecuciones del protocolo, y cada dato registrado en el tiempo es el siguiente en el diagrama.

Por último, y como un paso más en la calidad de la gestión de la información, pueden existir protocolos cuyo efecto sea el cambio de datos concretos asociados a un estado o a un dato longitudinal. La motivación de estos protocolos es registrar también en el sistema los cambios producidos por errores en el registro de la información. Es decir, cambios que no existen en el mundo real, sino que se deben a errores en la manipulación de la información. Por ejemplo, en el caso de los análisis de un incidente puede ser necesario corregir un dato registrado en el análisis realizado (manteniendo el valor anterior para dejar evidencia del cambio). En el ejemplo de los análisis clínicos puede ocurrir una situación similar, donde un valor se ha registrado de forma incorrecta y necesite ser corregido manteniendo el cambio de dato como evidencia. En estos casos, esos datos serían a su vez datos longitudinales en sí mismos, ya que tendríamos registrada su propia evolución en el tiempo mediante los acontecimientos generados en las correspondientes ejecuciones de protocolos de corrección. En el caso concreto de estos cambios de datos, es conveniente la inclusión de un atributo 'causa' en el que quede reflejado el motivo que ha provocado este cambio 'interno' de datos, mejorando la calidad de la gestión del sistema.

Capítulo 5. Una estrategia de diseño para el desarrollo de un SGBA basado en protocolos (SGBAP)

En el capítulo anterior, se ha presentado de forma general el concepto de acontecimiento basado en protocolos, así como el concepto de sistema de monitorización de procesos de negocio. En este capítulo se desciende a un nivel de detalle mucho mayor partiendo de los conceptos anteriores para describir una estrategia de diseño de un Sistema Gestor de Base de Acontecimientos basado en Protocolos –SGBAP-, es decir, de sistemas de monitorización de procesos utilizando las Bases de Acontecimientos basados en Protocolos como sus estructuras de persistencia.

Para ello se ha seguido una aproximación Model Driven Engineering –MDE- [11], mediante la que se pretende facilitar el proceso de construcción de un SGBAP. Con este objetivo, la estrategia proporciona una serie de patrones de diseño que ayudan en el proceso de representación del conocimiento específico de un dominio particular.

La estrategia consiste en tres fases:

- Definir los patrones de diseño que nos van a permitir modelar el concepto de acontecimiento basado en protocolos.
- Utilizar los patrones anteriores para desarrollar una metodología de construcción de Bases de Acontecimientos basados en Protocolos –BAP-.
- Definir la arquitectura de un SGBAP, que nos permitirá construir sistemas de monitorización utilizando estos conceptos.

Para representar estos patrones y conceptos se ha utilizado el estándar UML [69] de la siguiente forma:

- Diagramas de estados UML para representar los estados de los objetos (y sus cambios de estado).
- Diagramas de clase UML para representar las estructuras de persistencia de la BAP.

Es importante destacar que la estrategia de diseño presentada en este capítulo no es la única forma de construir SGBAPs, por lo que se podrían definir estrategias diferentes definiendo de forma diferente sus elementos (siempre que respeten las características de los acontecimientos basados en protocolos), como veremos en el Capítulo 8, o aplicando otras aproximaciones distintas a la MDE.

5.1 Perfil y patrones de acontecimientos basados en protocolos

En primer lugar, se ha definido un perfil UML [69][1] –el 'perfil de acontecimiento'- que nos permite extender la sintaxis UML a nuestro dominio de trabajo de acontecimientos, con el objetivo de particularizar sus conceptos respetando la semántica original del estándar. Los mecanismos que se utilizan para ello son:

- Estereotipos: Los estereotipos sirven para crear nuevos tipos de elementos a partir de otros ya existentes en el metamodelo UML o en otro perfil UML. Al aplicar un estereotipo sobre un elemento definimos su categoría. Su notación es entre los símbolos <<stereotype>>. En un estereotipo se pueden determinar propiedades, que definirán los atributos que tendrán los objetos que sean de esa clase.
- Restricciones: Determinan las condiciones a cumplir por un modelo para ser considerado válido. Su notación es entre llaves: {condición}.

Las razones para definir un perfil UML son variadas, como por ejemplo:

- Obtener una terminología y vocabulario específicos para un dominio de aplicación concreto. Como es nuestro objetivo.
- Ampliar la semántica de un metamodelo o perfil existente.
- Incluir restricciones específicas, impidiendo determinadas especificaciones que se podrían realizar en el modelo en otro caso.

Al definir nuestro perfil UML podemos, por tanto, representar más fácilmente el concepto de acontecimiento al introducir semántica de nuestro dominio en nuestros diseños:

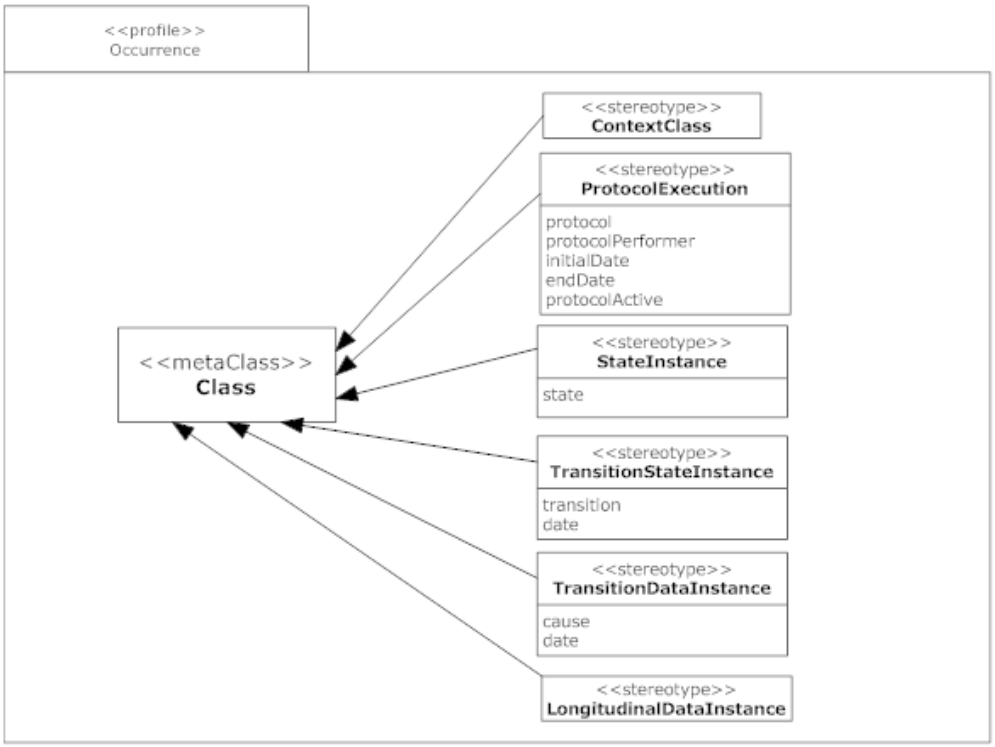


Figura 5-1 Perfil de acontecimiento

En el perfil se definen seis estereotipos, que extienden al elemento Clase de UML:

- ContextClass. Representa a los diferentes objetos que formarán parte de la estructura.
- StateInstance. Representa a los estados en los que podrá encontrarse un objeto.
- TransitionStateInstance. Representa las transiciones entre estados que pueden producirse por la ejecución de los protocolos.
- LongitudinalDataInstance. Representa los datos longitudinales asociados a los objetos, o los estados de los objetos, que son resultado de protocolos ejecutados.
- TransitionDataInstance. Representa las transiciones que se producen en los datos longitudinales por ejecución de los protocolos, también llamadas transiciones internas ya que se tratan de cambios de datos sin que haya un cambio de estado del objeto.
- ProtocolExecution. Como su nombre indica, representa las ejecuciones de protocolo que producen los acontecimientos en el sistema.

En estos estereotipos se definen los atributos que como mínimo deben tener las clases que los implementen para considerar que sus instancias son elementos de dichos tipos.

El estereotipo 'ProtocolExecution' contiene los atributos que permiten conocer quién ha ejecutado el protocolo –protocolPerformer-, información sobre el protocolo –protocol-, tiempo que ha llevado la ejecución –initial y end date-, y un atributo que establece si el protocolo está activo o no en el sistema –protocolActive- y que será crucial para permitir la evolución de los protocolos en el sistema (Capítulo 6).

En los estereotipos de transiciones, tanto de estado como de datos, se incluye el atributo que permite almacenar cuándo se ha producido dicha transición. En la transición de datos, además, el atributo 'cause' se añade para que se introduzca a través de él la causa que ha motivado dicho cambio de datos –por ejemplo, una corrección-, como medida de calidad del sistema.

Por último, en el caso de los estereotipos de transiciones 'StateInstance' y 'TransitionStateInstance', se incluyen los atributos 'state' y 'transition' al utilizar los diagramas de estados UML como mecanismos para definir el dinamismo del sistema, de los que son instancias respectivamente.

Estos estereotipos representan los conceptos principales que existen en el acontecimiento basado en protocolos, tanto para los casos en los que el efecto del acontecimiento sea un cambio de estado como para el caso que sea un cambio de datos asociado, lo que nos va a permitir modelar el concepto de acontecimiento dentro de nuestro dominio.

Para construir el concepto de acontecimiento basado en protocolos, necesitamos diseñar cinco patrones de diseño, que llamaremos patrones de acontecimientos, para contemplar los posibles efectos de las ejecuciones de protocolo:

- Patrón de Acontecimientos de Cambios de Estado. Patrón que permite diseñar estructuras de persistencia para registrar los cambios de estado producidos por la ejecución de un protocolo.
- Patrón de Acontecimientos de Creación de Objeto. Variante del patrón anterior que permite almacenar la creación de un objeto en su estado inicial.
- Patrón de Acontecimientos de Registro de Datos de Objeto. Patrón que permite diseñar

el almacenamiento de datos asociados a un objeto generados por la ejecución de un protocolo.

- Patrón de Acontecimientos de Cambio de Datos de Objeto. Patrón que define la estructura para registrar un cambio de datos de objeto por la ejecución de un protocolo.
- Patrón de Acontecimientos de Cambio de Datos de Estado. Patrón similar al anterior, en el que se define la estructura para registrar un cambio de datos asociado a un estado del objeto por acción de un protocolo.

Estos patrones nos van a permitir utilizar el concepto de acontecimiento basado en protocolos como un elemento atómico en estructuras más complejas, como las bases de acontecimientos basados en protocolos.

A continuación veremos estos patrones en más detalle. Para mayor facilidad en la comprensión de los diseños, se han utilizado diferentes colores [64] en función de la componente del acontecimiento al que refieren: objetos en rojo, efectos (estados, transiciones y datos) en verde y protocolos en azul.

5.1.1 Patrón de Acontecimientos de Cambios de Estado

En este patrón, como su propio nombre indica, se diseña la estructura de persistencia que se necesita para almacenar el cambio de estado que provoca en un objeto la ejecución de un protocolo, es decir, la información que conforma un acontecimiento basado en protocolos cuando su efecto es una transición entre estados:

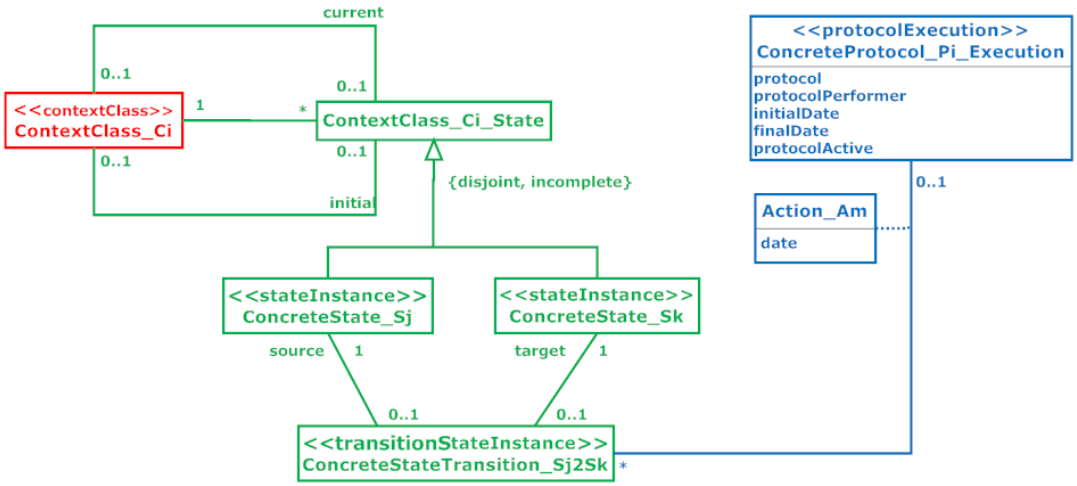


Figura 5-2 Patrón de Acontecimientos de Cambios de Estado

Para modelar este patrón de acontecimientos, se ha utilizado como referencia uno de los patrones de diseño de comportamiento del desarrollo de software, el patrón State [34]:

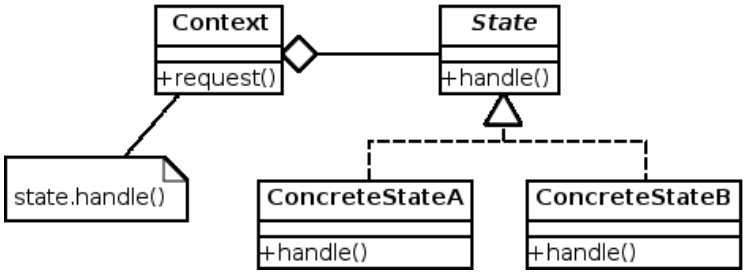


Figura 5-3 Patrón de Comportamiento State

El patrón de diseño State se utiliza cuando se necesita modelar que el comportamiento de un objeto cambia según su estado, lo que encaja perfectamente en el problema de diseño de nuestro caso.

Si analizamos ahora el patrón de acontecimientos, podemos observar que se encuentran los elementos 'ContextClass_Ci', en rojo, cuyas instancias corresponderán con las clases de objetos del sistema, y que corresponde al 'Context' del patrón State. En verde, se representa la estructura que seguirán los estados de los objetos. Por un lado, está el elemento general 'ContextClass_Ci_State' (elemento State del patrón de diseño) que abstrae como un elemento común todos los estados concretos del objeto (elementos ConcreteState). Esta abstracción nos permite además incluir dos relaciones a través de las que se puede conocer directamente el estado inicial y actual de un objeto, las cuáles serán muy útiles para la explotación posterior de la información, por ejemplo para la obtención de la línea de vida de un objeto (concepto definido en el apartado 3.3). Como hijos del estado abstracto, tenemos los estados concretos del objeto, que están representados de forma genérica en dos posibles elementos 'ConcreteState_Sj', 'ConcreteState_Sk'.

Todos los elementos anteriores son los que se identifican con el patrón State, sin embargo, necesitamos añadir más elementos que nos van a permitir incluir la información de las ejecuciones de protocolo. En primer lugar, a la derecha en azul, se encuentra la representación de forma genérica de la ejecución de protocolo 'ConcreteProtocol_PI_Execution', con los atributos que permiten almacenar la información del ejecutor de protocolo y las marcas de tiempo de su ejecución, así como el elemento 'Action' que representa la acción concreta que se realiza sobre el objeto.

El elemento 'Action' tiene su propia marca de tiempo diferenciada de las marcas de la ejecución de protocolo. Esto es debido a que la ejecución del protocolo puede extenderse en el tiempo, y el cambio de estado producirse en un momento concreto dentro de esa ejecución. Por ejemplo, pensemos en la ejecución de un protocolo que provoca el cambio de estado de dos objetos (generando dos acontecimientos). El primer cambio de estado puede no ocurrir en el mismo instante que el segundo, y la ejecución de protocolo englobará los dos cambios de estado. Esta doble consideración del tiempo de la ejecución del protocolo y de la ocurrencia del efecto del acontecimiento está presente en todos los patrones de acontecimientos.

Por último, tenemos el efecto que la ejecución del protocolo provoca en el objeto, el elemento 'ConcreteTransition_Sj2Sk', que representa el posible cambio de estado del estado Sj al estado Sk. En este caso es importante destacar las cardinalidades establecidas en la asociación de la ejecución de protocolo y la transición entre estados. Como se ve en el diseño

del patrón, una ejecución de protocolo puede disparar varias transiciones de estado, mientras que la transición de estado puede estar asociada a una o ninguna ejecuciones de protocolo.

La explicación de la primera cardinalidad (una ejecución dispara varias transiciones) es sencilla, ya que en una ejecución de protocolo se pueden realizar varias transiciones de estado porque afecte a varias instancias de objeto distintas al mismo tiempo. Por ejemplo, en el protocolo de almacenamiento de muestras se pueden almacenar varias muestras al mismo tiempo, por lo que habrá tantas transiciones de estado generadas por la ejecución del protocolo como muestras se almacenen.

La segunda cardinalidad (una transición está asociada a una o ninguna ejecución de protocolo) se establece por la necesidad de dar generalidad al patrón de diseño, y poder adapatarlo a diferentes situaciones que pueden producirse. Es el caso, por ejemplo, de una transición de estados que pueda producirse por la ejecución de dos protocolos diferentes. La transición estará asociada a una de las dos ejecuciones de protocolo, y no lo estará a la otra, por lo que en términos de diseño se requiere la cardinalidad '0..1'.

En todos los patrones que describimos en este capítulo, aparecen cardinalidades similares a las que acabamos de comentar. Para no se repetitivos, en los demás patrones no volvemos a repetir estas explicaciones.

Si analizamos el patrón anterior, podemos observar claramente los elementos que conforman la tupla de un acontecimiento basado en protocolos (ep, o, d): la ejecución de protocolo –ep-, el objeto –o-, y los estados y el cambio de estado respectivamente –d-.

En este patrón se han incluido los elementos principales que definen la estructura del acontecimiento, obviando otro tipo de aspectos, como datos estáticos del objeto o relaciones entre objeto, que pueden existir en un sistema real pero que no condicionan el propósito del patrón. Al diseñar una base de acontecimientos concreta, estos datos se deben añadir para completar la información básica del patrón.

Al aplicar este patrón a un caso concreto, obtenemos una estructura que nos permite almacenar, para cada objeto del sistema, las ejecuciones de protocolo que le han provocado cambios de estado. Por ejemplo, en la siguiente figura se muestra la aplicación del patrón anterior al caso de un protocolo para el almacenamiento de muestras biológicas.

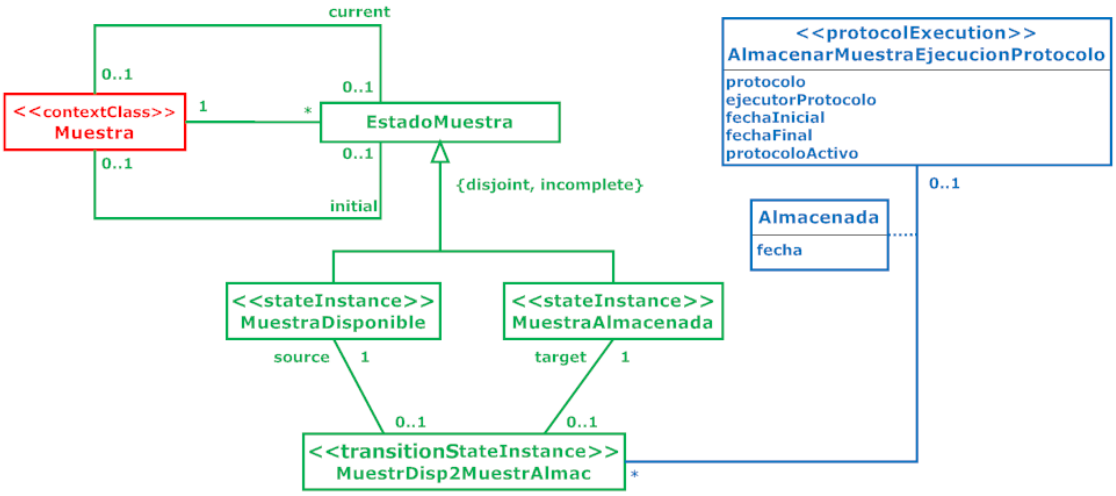


Figura 5-4 Ejemplo de aplicación del Patrón de Acontecimientos de Cambios de Estado

En este ejemplo, el objeto muestra tiene dos posibles estados, disponible y almacenada, y el cambio de estado entre ellos lo realiza la ejecución del protocolo de almacenamiento, registrándose la acción de almacenada.

5.1.2 Patrón de Acontecimientos de Creación de Objetos

El patrón anterior es la estructura genérica que representa la gestión de los estados de un objeto, sin embargo, también necesitamos definir un caso especial que es la creación del objeto en sí, en la que no hay un cambio de estado propiamente dicho ya que no hay un estado anterior de partida (el objeto no existe). Este caso se define en el 'Patrón de Acontecimientos de Creación de Objetos':

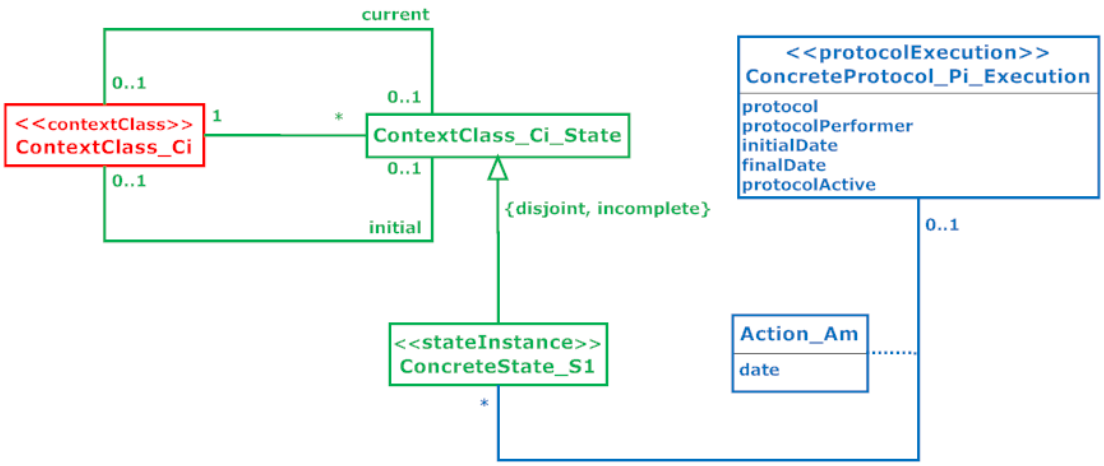


Figura 5-5 Patrón de Acontecimientos de Creación de Objetos

Como se puede ver en la imagen, el patrón es prácticamente igual que el de cambio de estado, con la salvedad de que en este caso sólo hay un estado (el inicial), y que en este caso el protocolo de ejecución está relacionado directamente con el propio estado al no haber un elemento de cambio de estado. Veámoslo con un ejemplo, esta vez relativo a la gestión de incidencias con el protocolo de registro de un nuevo incidente:

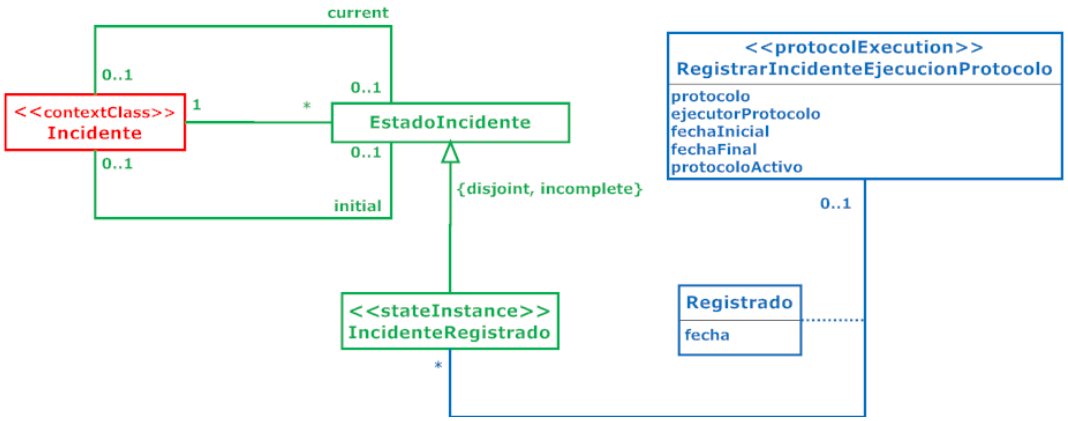


Figura 5-6 Ejemplo de aplicación del Patrón de Acontecimientos de Creación de Objeto

En el ejemplo anterior, cada vez que se ejecuta el protocolo de registro del incidente se crea una instancia del objeto incidente en su estado inicial 'registrado'.

5.1.3 Patrón de Acontecimientos de Registro de Datos de Objeto

Los patrones anteriores nos han permitido diseñar las estructuras que refieren a los efectos relacionados con los estados de un acontecimiento basado en protocolos. Sin embargo, como hemos comentado anteriormente, en este tipo de acontecimientos el efecto también puede ser el registro de datos longitudinales asociados al objeto, los cuáles nos van a permitir conocer la evolución en el tiempo de la información del objeto.

El Patrón de Acontecimientos de Registro de Datos permite diseñar estructuras de persistencia para el caso en el que el efecto del acontecimiento basado en protocolos es el almacenamiento de datos asociados al objeto:

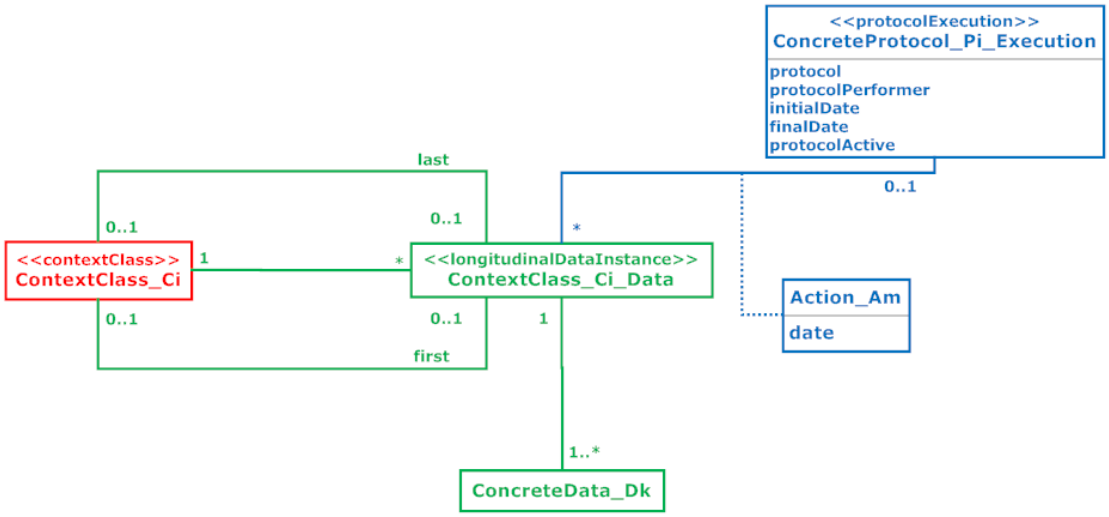


Figura 5-7 Patrón de Acontecimientos de Registro de Datos de Objeto

Recordemos que en el apartado 4.2.3 explicábamos la naturaleza de los datos longitudinales que se registran asociados a un objeto, siendo ésta similar a la información de los estados del objeto. La diferencia radicaba en que en vez de tener un diagrama de estados concreto que define un conjunto finito y ordenado de estados, tenemos una estructura en forma de infinito potencial, en la que el potencial de datos registrados longitudinalmente sobre el objeto puede ser infinito. Mediante el diseño anterior, el objeto tendrá tantos datos asociados como ejecuciones del protocolo hayan sucedido (potencialmente infinitas). Los datos son longitudinales por la forma en que se determinan, ya que están supeditados a la ejecución de un protocolo en un momento determinado del tiempo, por lo que se puede establecer en ellos una cronología.

El dato longitudinal se ha estructurado como un objeto en sí mismo, con su clase 'ContextClass_Ci_Data' que lo identifica como dato asociado al objeto 'ContextClass_Ci', y tantos elementos 'ConcreteData_Dk' como conjuntos de subdatos distintos conformen el dato longitudinal. Por ejemplo, pensemos en el registro longitudinal de análisis de sangre sobre un paciente. El análisis de sangre en abstracto será el 'ContextClass_Ci', pero éste está compuesto de muchos tipos diferentes de datos concretos (colesteroles, leucocitos, plaquetas…).

Si analizamos el patrón, podemos identificar los elementos del acontecimiento basado en protocolos (ep, o, d): la ejecución de protocolo –ep-, el objeto –o-, y el dato longitudinal como efecto del acontecimiento –d-.

Al igual que en los patrones de estado, se han incluido las relaciones que permiten obtener los primeros y últimos datos registrados al objeto, con el objetivo de proporcionar eficiencia en la gestión longitudinal de la información. En este caso no podemos hablar de datos iniciales y actuales como en los estados, ya que en los datos longitudinales el concepto de actualidad refiere al último dato registrado en el tiempo, lo que puede no ser sinónimo de actualidad de una información. Por ejemplo, siguiendo con el registro en el tiempo de los análisis de sangre realizados a una persona. El último análisis registrado sobre esa persona no debe entenderse como la actualidad de esa información, ya que puede ser un análisis registrado hace un mes que evidentemente no es la actualidad de esa persona. En la siguiente imagen puede verse la utilización de este patrón para el diseño de este ejemplo de registro de analíticas:

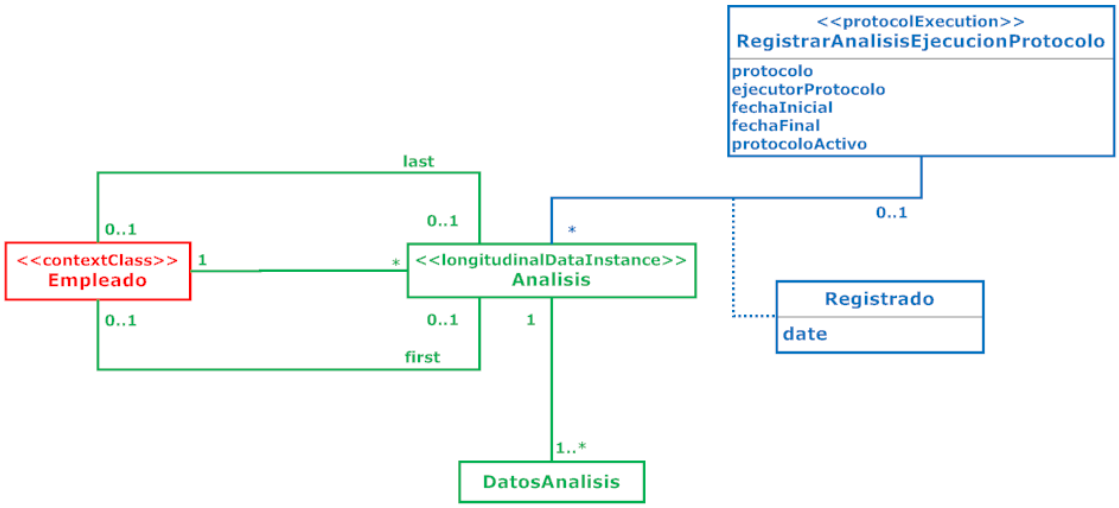


Figura 5-8 Ejemplo de Patrón de Acontecimientos de Registro de Datos de Objeto

5.1.4 Patrón de Cambio de Datos de Objeto

Este patrón define la estructura para registrar los cambios que se producen en los datos longitudinales asociados a un objeto, siendo su diseño el siguiente:

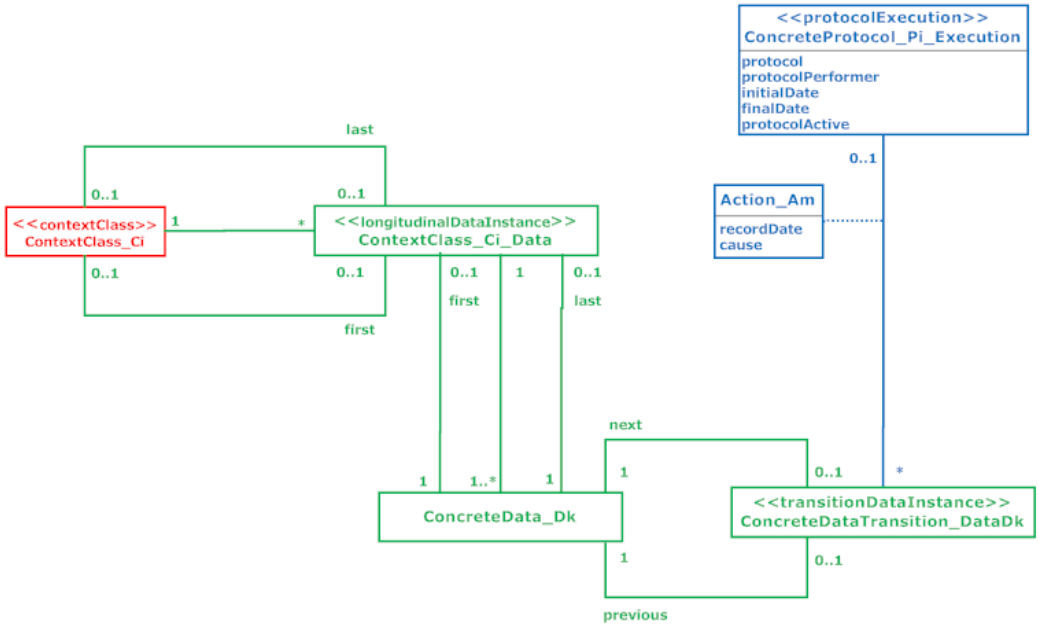


Figura 5-9 Patrón de Acontecimientos de Cambio de Datos de Objeto

El patrón parte de una estructura similar al del registro de datos del objeto, con el objeto 'ContextClass_Ci ' en rojo a la izquierda y a la derecha, en azul, la representación de la ejecución de protocolo 'ConcreteProtocol_PI_Execution', con los atributos del ejecutor de protocolo y de tiempo de su ejecución, así como el elemento 'Action' que representa la acción concreta que se realiza sobre los datos concretos del dato longitudinal. A diferencia del patrón anterior, en la acción se han incluido los atributos que permiten registrar el instante concreto en el que se registra en el sistema el acontecimiento y la causa que ha motivado dicho cambio de datos.

Además, también se recoge el efecto que la ejecución del protocolo provoca en el objeto, el elemento 'ConcreteDataTransition_Di', que representa el cambio que se produce en los datos concretos del dato longitudinal asociado al objeto. La transición es una relación reflexiva entre los propios datos concretos, lo que va a permitir registrar esta 'longitudinalidad' derivada en el propio dato longitudinal, en la que sus datos concretos pueden ser a su vez datos longitudinales por sí mismos.

En este caso, al igual que en los patrones anteriores, tenemos todos los elementos que conforman un acontecimiento basado en protocolos (ep, o, d): la ejecución de protocolo –ep-, el objeto –o- (al que está asociado el dato longitudinal), y los nuevos datos que se registran en el dato longitudinal –d-.

Un ejemplo de este patrón lo podemos encontrar en la inclusión de sistemas de corrección de errores en el registro de datos como, por ejemplo, para registrar las modificaciones realizadas en los datos de un reconocimiento médico para corregir respuestas en el cuestionario de salud que se han introducido erróneamente en el sistema.

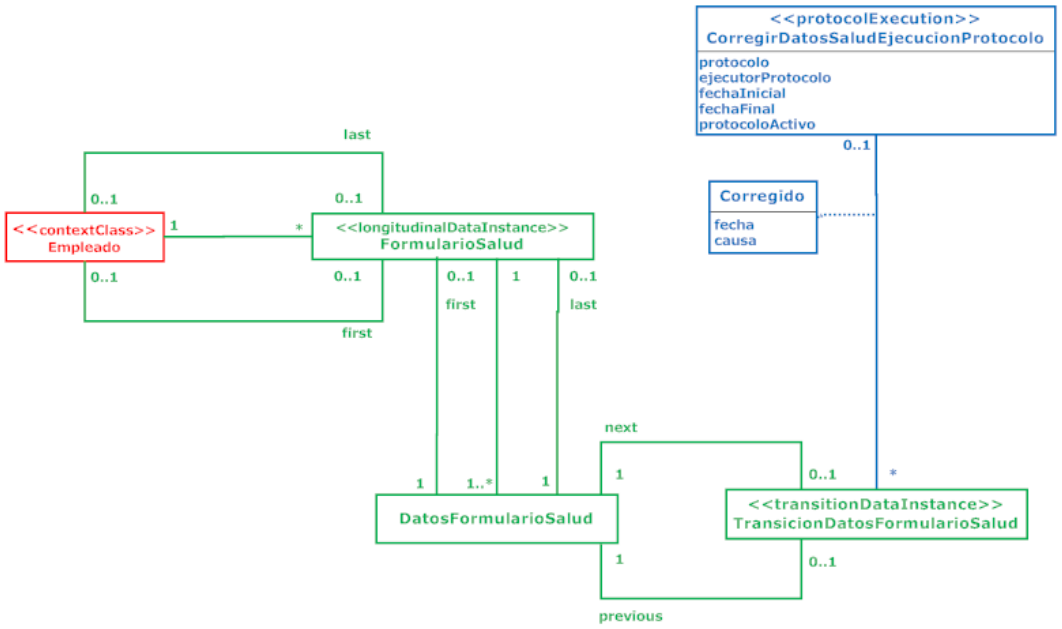


Figura 5-10 Ejemplo de Patrón de Acontecimientos de Cambio de Datos de Objeto

5.1.5 Patrón de Cambio de Datos de Estado

Este patrón define la estructura para registrar los cambios que se producen en los datos longitudinales asociados a un estado del objeto:

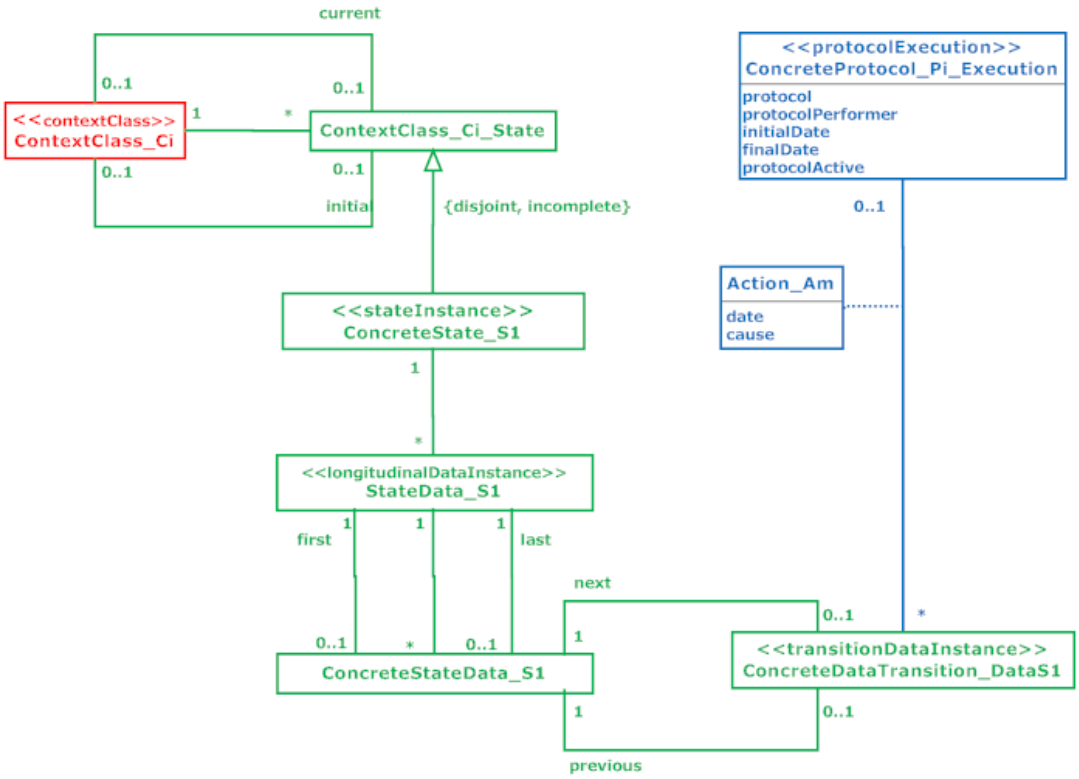


Figura 5-11 Patrón de Acontecimientos de Cambio de Datos de Estado

Como se puede ver, la variante que introduce este patrón en los estados de un objeto tiene una estructura similar a los patrones de datos anteriores. En concreto, los datos asociados al estado se tratan como un dato longitudinal, en el que la ejecución del protocolo 'ConcreteProtocol_Pi_Execution' provoca (elemento 'ConcreteDataTransition_DataS1') el cambio del dato antiguo al nuevo. La transición, al igual que en el patrón de cambio de datos anterior, es una relación reflexiva que va a permitir registrar la relación 'siguiente-anterior' de los datos.

En el caso de los datos longitudinales asociados al estado del objeto, su creación se produce en el momento en el que se registra el estado, estando asociado por tanto al protocolo de creación o de cambio de estado según sea el caso, ya que son datos inherentes a que el objeto esté en dicho estado. Por este motivo, en el caso de los datos asociados a un estado del objeto únicamente es necesario el diseño del cambio de datos.

Al igual que en todos los patrones anteriores, podemos identificar los elementos del acontecimiento basado en protocolos (ep, o, d): la ejecución de protocolo –ep-, el objeto –o- (al que está asociado el estado en el que se registran los datos), y los nuevos datos que se registran en el estado –d-.

Un ejemplo claro de aplicación de este patrón lo encontramos en el protocolo de análisis en la gestión de incidencias. En este caso, los datos que se gestionan en esta actividad requieren que estén asociados al estado 'analizado' del incidente, ya que un incidente puede sufrir varios análisis en su resolución debido que el incidente se escala a otros equipos técnicos, o debido a que la solución propuesta en el análisis no sea correcta y necesite ser re-analizado. Además, los datos registrados para un análisis requieren que puedan ser modificados para incluir progresivamente toda la información que se obtiene de diferentes estudios, pruebas, etc., que pueden ser necesarias para conocer la causa del incidente. En la siguiente imagen puede verse la aplicación de este patrón a este caso:

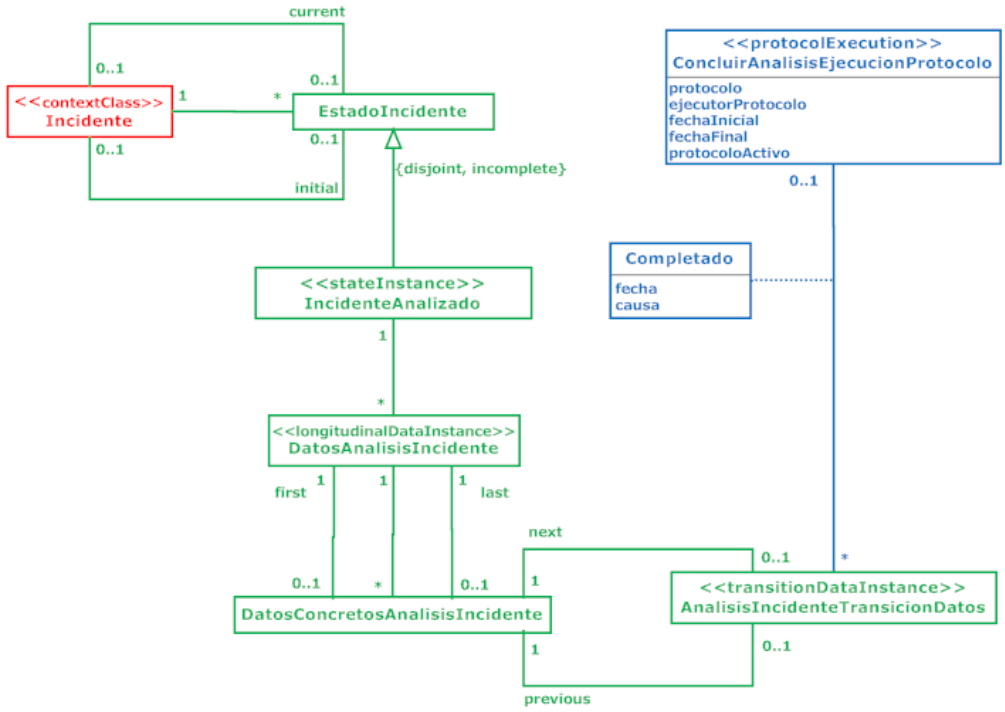


Figura 5-12 Ejemplo de Patrón de Acontecimientos de Cambio de Datos de Estado

5.2 Una metodología para el desarrollo de Bases de Acontecimientos basados en Protocolos

Tomando como partida los patrones de acontecimientos diseñados en el apartado anterior, el siguiente paso en la estrategia de diseño de un SGBAPs es la definición de una metodología para el desarrollo de Bases de Acontecimientos basados en Protocolos. Esta metodología va a ayudar en el proceso de construcción de la la estructura de persistencia que utilizará el sistema de monitorización de procesos.

La metodología se ha desarrollado utilizando una aproximación MDA (Arquitectura Dirigida por Modelos, Model-Driven Architecture –MDA- en inglés [68]). La aproximación MDA consiste en definir primero un modelo independiente de plataforma (Platform Independent Model, -PIM-) en el que se define la estructura de funcionamiento de un sistema sin las particularidades de una plataforma de implementación específica. Posteriormente, se transforma el modelo PIM a las particularidades de de la plataforma concreta elegida para su implementación, desarrollando los llamados modelos específicos de plataforma (Platform Specific Model -PSM-). Para cada plataforma diferente en la que se implemente el sistema se define un modelo PSM propio.

Para representar los elementos de esta aproximación se han utilizado:

- Diagramas de clase UML estereotipados con el perfil de acontecimientos, definido en el anterior, para la especificación de un patrón para el desarrollo de los modelos independientes de plataforma –PIM-.
- Diagramas de clase UML no estereotipados para la especificación de un patrón para el desarrollo de los modelos específicos de plataforma –PSM-.

En el caso de la metodología presentada en este apartado, como veremos a continuación, el patrón PSM se ha desarrollado tomando como referencia las bases de datos relacionales como plataforma de implementación de una Base de Acontecimientos basado en Protocolos.

5.2.1 Patrón de Modelos Independientes de Plataforma –Patrón PIM-

Los modelos PIM son los modelos que representan a nivel conceptual las características del sistema, definiendo su alcance, los objetos de negocio, sus comportamientos en el sistema y las relaciones entre ellos. Al ser un diseño conceptual, el modelo resultante no depende de los requisitos de una plataforma de implementación específica, de ahí su nombre.

En el caso de la estrategia de diseño, este tipo de modelos los vamos a utilizar para definir la estructura de la Bases de Acontecimientos basados en Protocolos del sistema, diseñando todos los protocolos que actúan sobre los objetos de nuestro sistema y que van a producir los acontecimientos.

Para facilitar el desarrollo de este modelo, en la metodología de construcción de las bases de acontecimientos se define el patrón PIM. Este patrón va a permitir construir el modelo PIM del sistema objetivo mediante la repetición de estructuras que modelan los acontecimientos basados en protocolos (los patrones de acontecimientos definidos en el apartado anterior). En la siguiente figura se puede observar el patrón de modelos PIM diseñado para desarrollar Bases de Acontecimientos basados en Protocolos –Patrón PIM de Base de Acontecimientos-:

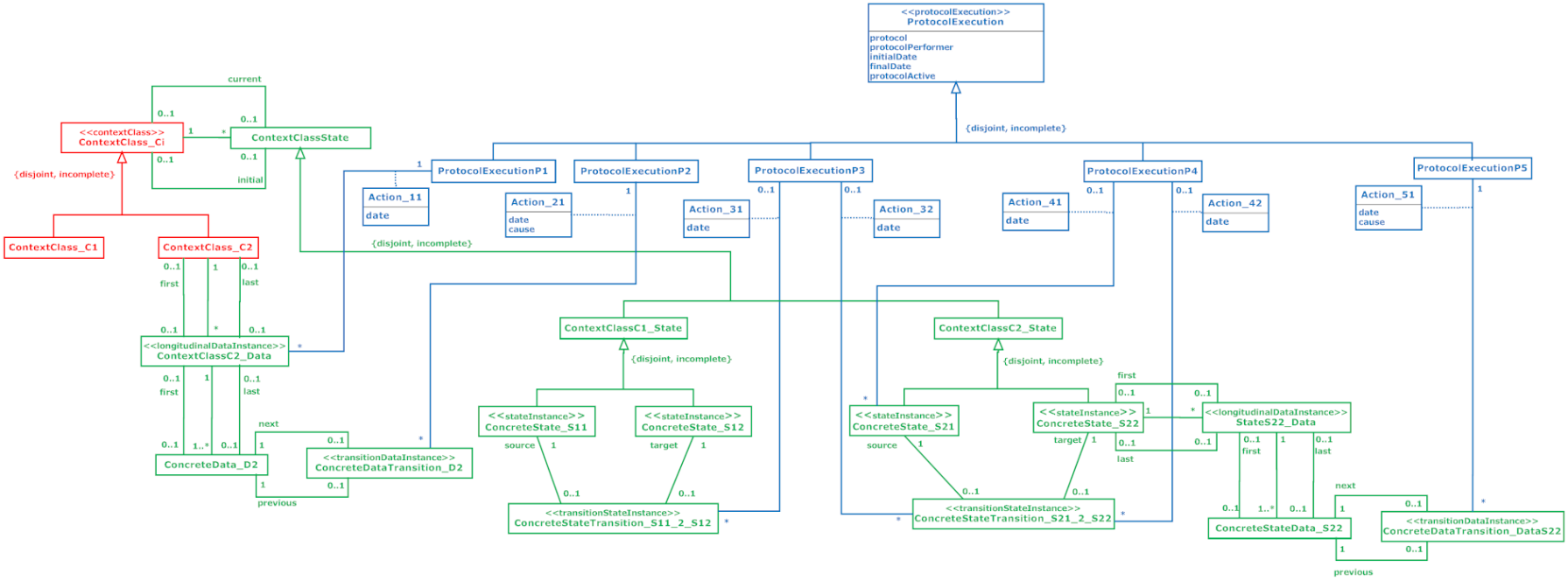


Figura 5-13 Patrón PIM de Base de Acontecimientos

En los elementos del patrón PIM se han utilizado los estereotipos definidos en el Perfil de Acontecimiento para incluir la semántica de nuestro dominio.

El patrón está construido utilizando como base los cinco patrones de acontecimientos definidos en el apartado anterior. Para construir el modelo PIM de un sistema en particular, los patrones de acontecimientos se utilizarán a modo de bloques repetitivos para todos los protocolos y estados diferentes que existan en los objetos a considerar (establecidos mediante los correspondientes diagramas de actividad y estados UML que se hayan desarrollado en el análisis del sistema).

La utilización de todos los patrones en el desarrollo de un modelo PIM concreto no es de cumplimiento obligatorio, sino que dependerá de cada sistema en particular. Por ejemplo, en un sistema puede no ser necesaria la consideración de acontecimientos de datos longitudinales, utilizándose en ese caso únicamente los patrones de estados; o puede ser necesaria sólo la consideración, dentro de los acontecimientos de datos longitudinales, del patrón de Registro de Datos sin incluir los patrones de cambios de datos.

Para ilustrar la forma de aplicar los patrones de acontecimientos, en la Figura 5-13 se muestra cómo se utilizan dichos patrones tomando como referencia varias ejecuciones de protocolo diferentes.

- El Patrón de Registro de Datos se ha utilizado en la ejecución de protocolo 'ProtocolExecutionP1', donde se registran los datos longitudinales asociados al objeto 'ContextClass_C2'.
- El Patrón de Cambio de Datos de Objeto se ha representado en la ejecución de protocolo 'ProtocolExecutionP2', en el cuál el efecto de este protocolo es el cambio de los datos asociados al objeto 'ContextClass_C2' (que habrán sido creados en primera instancia con la ejecución del protocolo anterior).
- El Patrón de Acontecimiento de Cambio de Estado lo podemos observar en la ejecución de protocolo 'ProtocolExecutionP3', que provoca el cambio de estado 'ConcreteTransition_S11_2_S12'.
- El Patrón de Acontecimientos de Creación de Objetos por su parte, lo podemos ver en el ejemplo de 'ProtocolExecutionP4', estando asociado con el estado 'ConcreteState_S21' de la creación del objeto 'ContextClass_C2'.
- El Patrón de Cambio de Datos de Estado, por último, se ha representado mediante la ejecución del protocolo 'ProtocolExecutionP5', cuyo efecto consiste en el cambio de los datos longitudinales registrados para el estado 'ConcreteState_S22'.

Con el objetivo de una mejor organización de los estados de los objetos, se han incluido las clases 'ContextClassCn_State' como elemento padre de todos los estados que puede tener un objeto de la clase 'ContextClass_Cn' correspondiente. Tanto los objetos como sus estados están organizados mediante dos IS-A generales, 'ContextClass_Ci' y 'ContextClassState' respectivamente, que proporcionan una asociación común entre los objetos y los estados.

De la misma forma, todos los protocolos concretos definidos en el sistema tienen como elemento padre la clase 'ProtocolExecution', donde están definidos todos los atributos comunes de los protocolos, como el ejecutor del protocolo y los tiempos en los que se produce la ejecución del protocolo. Los hijos de esta IS-A serán cada uno de los diferentes protocolos

que existirán en el sistema.

La inclusión de estas clases más generales otorga una menor complejidad en el tratamiento de los acontecimientos, al poder utilizar elementos comunes para su gestión y facilitando por tanto su explotación.

Por otro lado, en el patrón PIM se incluye el hecho de que un protocolo puede causar efectos en objetos de diferentes clases de contexto. Esto se ha representado en la ejecución de protocolo 'ProtocolExecutionP3', la cual está relacionada con dos cambios de estado de los objetos 'ContextClass_C1' y 'ContextClass_C2', concretamente con 'ConcreteTransition_S11_2_S12' y 'ConcreteTransition_S21_2_S22' respectivamente. Un ejemplo de este tipo se produce cuando se almacena la última muestra que cabe en una caja. En este caso, el protocolo provocará dos cambios de estado: el de la muestra para quedar en el estado 'almacenada' y el de la caja para quedar en el estado 'completa'.

También se modela el caso de que un objeto puede estar afectado por diferentes protocolos. En el patrón, el objeto 'ContextClass_C2' está afectado por los protocolos 'ProtocolExecutionP3' y 'ProtocolExecutionP4'. Por ejemplo, en el caso de la gestión de incidencias, el objeto incidente estará afectado por todos los protocolos que componen el proceso de gestión de la incidencia: el protocolo de registro (para la creación del objeto), el protocolo de clasificación (cuyo efecto será el que el incidente quede en estado 'clasificado'), el protocolo de análisis…

En la siguiente imagen se muestra un ejemplo de desarrollo del patrón PIM, mostrando un extracto del modelo PIM construido en el caso ARASIS en el apartado 7.1.1.

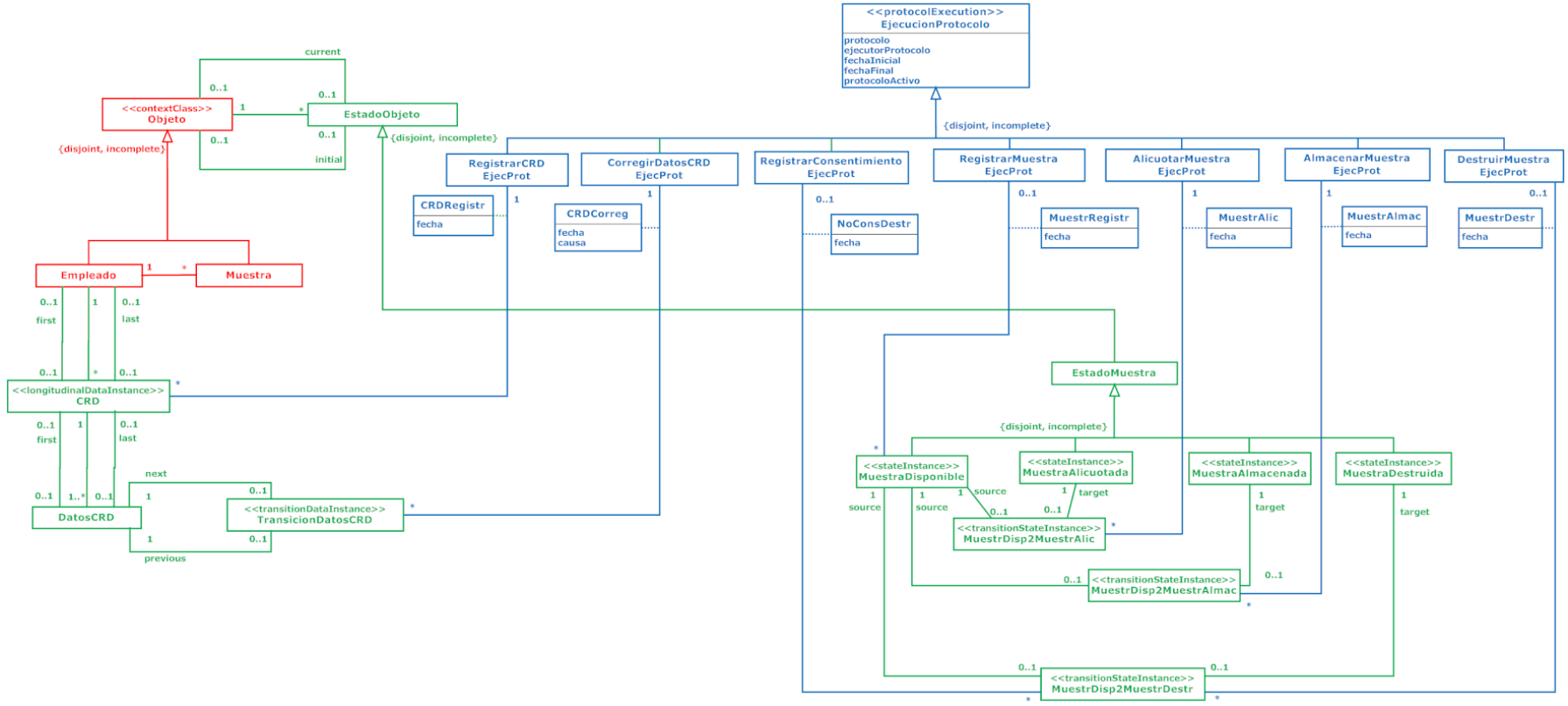


Figura 5-14 Ejemplo de modelo PIM de Base de Acontecimientos

5.2.2 Patrón de Modelos específicos de plataforma – patrón PSM-

El patrón PIM anterior nos proporciona el mecanismo para modelizar a nivel conceptual una base de acontecimientos basados en protocolos, sin considerar las condiciones particulares de la plataforma en la que va a ser desarrollada.

Una vez que se decide la plataforma de implantación, el siguiente paso es construir el modelo PSM, mediante la transformación del modelo PIM anterior a los requisitos de la plataforma objetivo.

En el caso de las bases de acontecimientos, como ejemplo a partir de ahora, vamos a suponer que se construirán como bases de datos relacionales en un SGBD como puede ser Oracle o PostgreSQL. Por tanto, el modelo PSM a desarrollar debe proporcionarnos un modelo lo más cercano posible a la base de datos relacional objetivo. Para ello, al igual que para construir el modelo PIM hemos definido un patrón de diseño, para el modelo PSM se ha definido el patrón PSM de Base de Acontecimientos transformando el patrón PIM anterior:

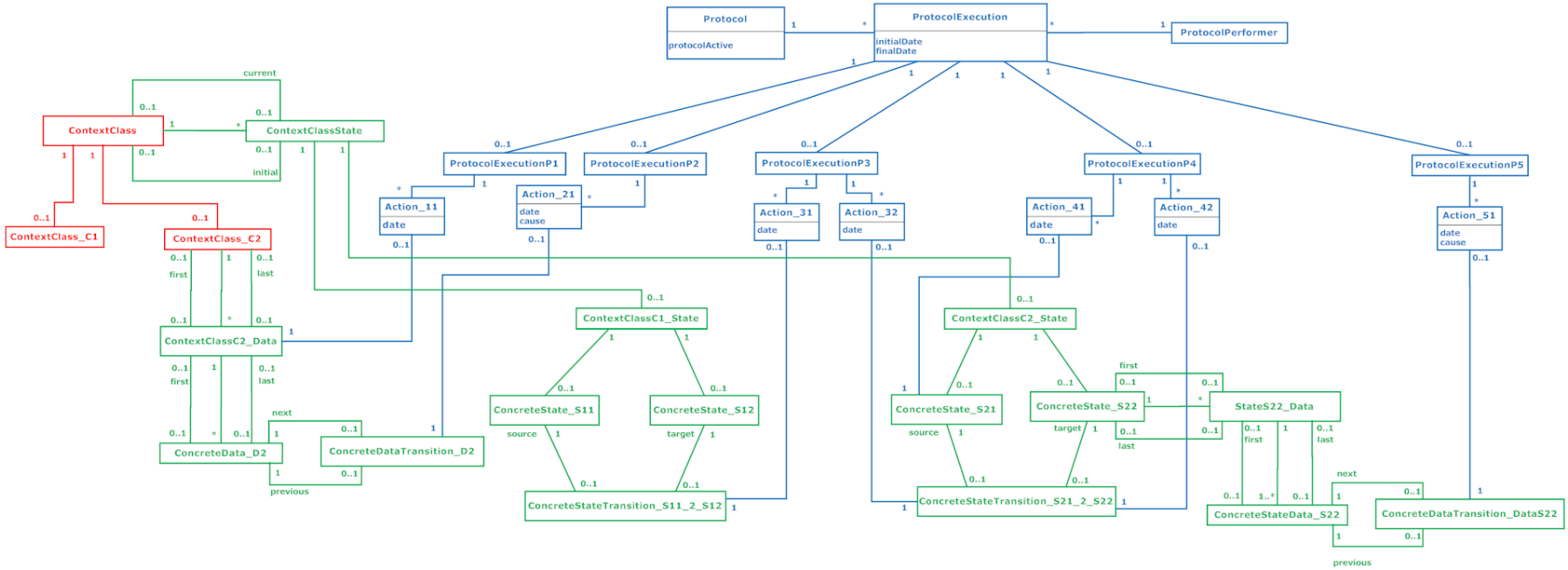


Figura 5-15 Patrón PSM de Base de Acontecimientos

En las transformaciones de modelos UML según la metodología MDA, se pueden aplicar dos técnicas [61]:

- Transformación endógena. Las transformaciones se aplican dentro del mismo metamodelo, en el mismo lenguaje.
- Transformación exógena. La transformación se realiza entre modelos que atienden a metamodelos diferentes.

La transformación que se ha realizado en este caso es exógena, ya que el patrón PSM hace uso de un metamodelo diferente al patrón PIM. En el caso del patrón PIM el metadomelo está esterotipado –perfilado, mediante un perfil UML-, con los estereotipos definidos en el perfil de acontecimiento, mientras que el metamodelo definido en el patrón PSM es más restrictivo y sin estereotipos –no perfilado-.

Al llevar a cabo la transformación, en primer lugar, una parte del PSM se ha obtenido utilizando los estereotipos de nuestro perfil como guía en su proceso de transformación, de acuerdo a la propuesta MDA de OMG [68]. En concreto, los estereotipos se han reemplazado por clases que almacenan la información relacionada con ellos. Es el caso, por ejemplo, de las clases de contexto o los estados de los objetos, los cuales eran elementos estereotipados en el patrón PIM que representaban ese tipo de elementos del acontecimiento, y que se convierten en clases de objetos equivalentes en el patrón PSM.

También se han transformado aquellos atributos definidos en los estereotipos que son composición de datos, que en el patrón PSM se han convertido en nuevas clases y asociaciones. Esta transformación es la que se ha realizado en los atributos de la ejecución del protocolo, donde se incluían el ejecutor de protocolo y la referencia al propio protocolo:

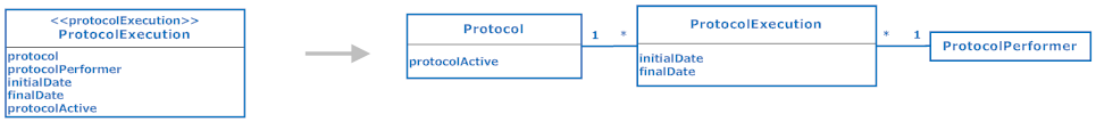


Figura 5-16 Ejemplo de transformación de atributos de composición

Al realizar la transformación, se definen las cardinalidades de las nuevas asociaciones con la clase 'ProtocolExecution'. Cada ejecución de protocolo estará asociada con un único protocolo (cuya ejecución representa) y con un único ejecutor de protocolo (que realiza la ejecución), mientras que un protocolo puede tener varias ejecuciones de protocolo, y un ejecutor de protocolo, como es obvio, puede realizar varias ejecuciones de protocolo.

Por otro lado, las clases asociativas definidas en el patrón PIM, las acciones, se han transformado en una nueva clase con dos asociaciones hacia los elementos que unían –ejecuciones de protocolo y transiciones por ejemplo-, de forma que se realizará correctamente la propagación de identificadores de objeto en la construcción de la base de datos relacional final.

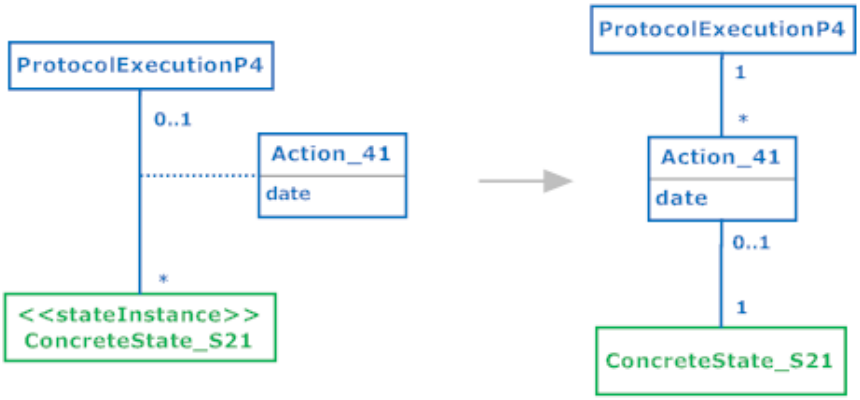


Figura 5-17 Ejemplo de transformación de clase asociativa

Esta transformación mantiene la naturaleza de la asociación original. Una instancia de la acción estará asociada únicamente a una ejecución de protocolo y, en este caso, al estado concreto que ha generado. El ejemplo anterior es similar a las transformaciones realizadas para las acciones de datos o de cambio de estado.

Por último, la transformación principal realizada en el patrón PSM es la sustitución de las relaciones de generalización, las IS-A, que existían en el patrón PIM, ya que este tipo de relaciones no se pueden implementar directamente en una base de datos relacional en su nivel lógico. Para la transformación de las IS-A se pueden utilizar dos tipos de transformaciones [28]:

- Transformación horizontal. En este tipo de transformación, en el modelo resultante únicamente se conservan los 'hijos' de la relación, perdiéndose la generalidad que proporciona el elemento 'padre' de la relación.
- Transformación vertical. Esta transformación consiste en definir una clase tanto para el padre como para los hijos de la relación.

En el caso del patrón PSM, se ha realizado una transformación vertical de las relaciones IS-A, tal y como se puede ver en el ejemplo de los estados:

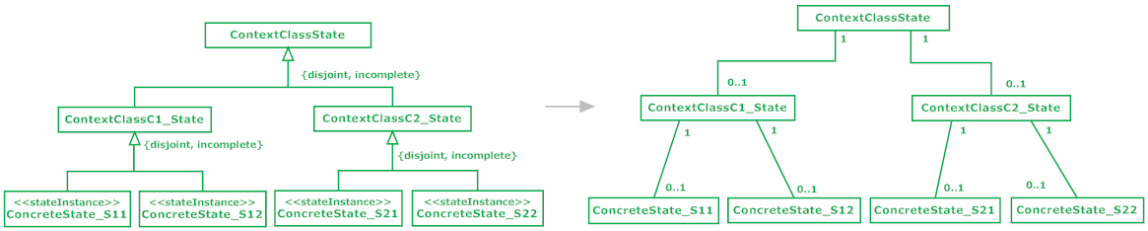


Figura 5-18 Ejemplo de transformación de relaciones de generalización

Para mantener las características de la relación IS-A, las cardinalidades resultantes en el diseño transformado han de incluirse de forma que todo hijo esté relacionado únicamente con su elemento padre –cardinalidad 1-, mientras que los elementos padres estarán o no relacionados con un único elemento hijo –cardinalidad 0..1-. Para mantener este criterio de

unicidad, en la implementación concreta del modelo PSM, en la base de datos relacional correspondiente, será necesario añadir restricciones de integridad –constraints- que lo aseguren.

El patrón PSM resultante de aplicar las transformaciones anteriores, nos permite obtener un modelo de objetos que cumple las restricciones de las bases de datos relacionales preparado para construir el esquema relacional en el SGBD de la base de datos objetivo (por ejemplo Oracle o PostgreSQL). En la siguiente imagen se muestra un ejemplo de desarrollo del patrón PSM, mostrando un extracto del modelo PSM construido en el caso ARASIS en el apartado 7.1.1.

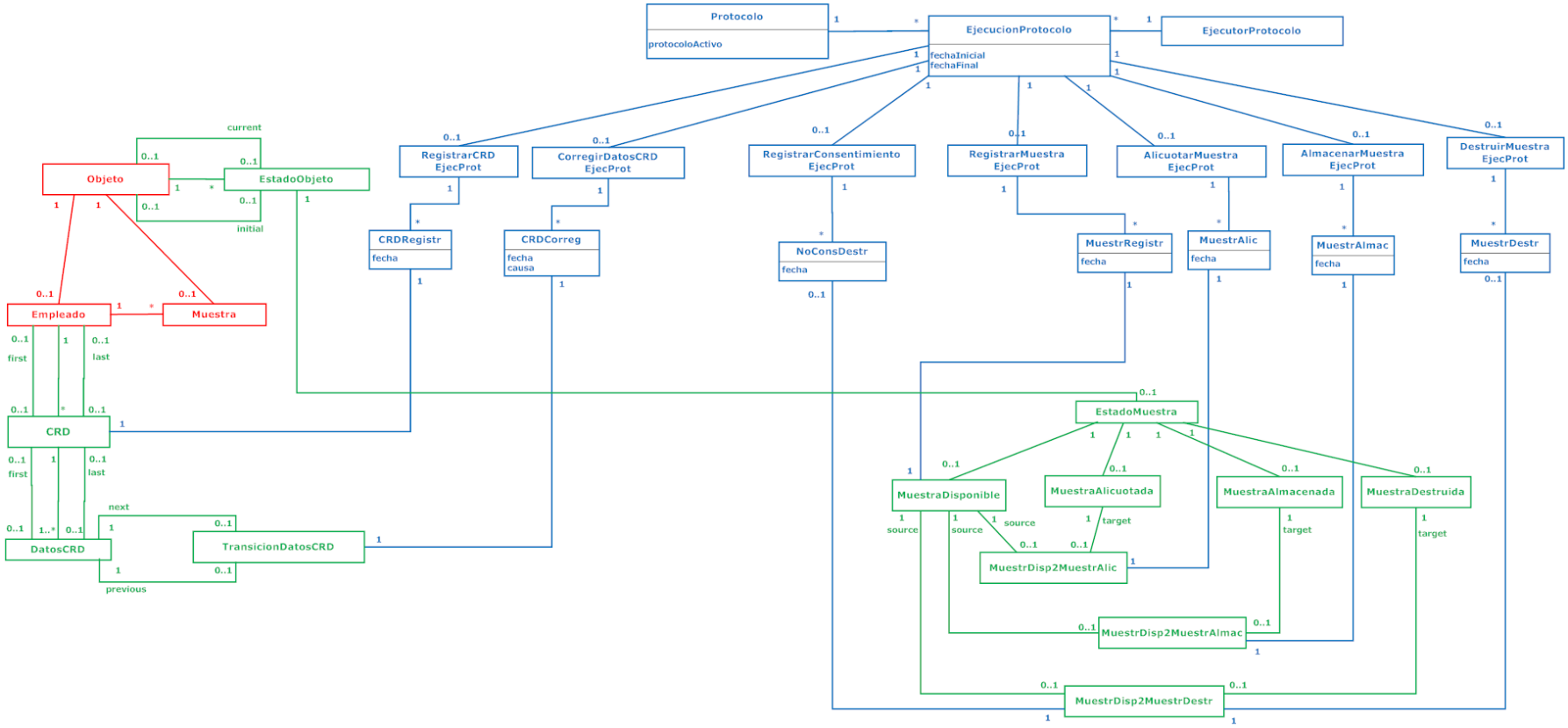


Figura 5-19 Ejemplo de modelo PSM de Base de Acontecimientos

5.3 Una arquitectura para el desarrollo de SGBAPs

Una vez desarrollada la metodología para construir la Base de Acontecimientos basado en Protocolos, que nos permitirá construir las estructuras de persistencia de la información que se genere en el sistema, el último paso en la estrategia de diseño es el desarrollo de una arquitectura que nos permita construir un Sistema Gestor de Bases de Acontecimientos basados en Protocolos –SGBAP- para explotar su conocimiento.

El principal objetivo de un SGBAP –al que llamaremos OcSystem en inglés- es facilitar la monitorización de una aplicación que implementa varios procesos de negocio, además de permitir la explotación del conocimiento de la Base de Acontecimientos –OcBase-.

La arquitectura propuesta para ello está basada en el patrón Model-View-Controller –MVC- [34]. El patrón MVC es un patrón de arquitectura de software cuyo objetivo es separar la gestión de datos, lógica de negocio e interfaz de usuario que conforman una aplicación de software, con el propósito de facilitar su desarrollo y posterior mantenimiento.

Para ello, el patrón MVC propone la construcción de tres componentes:

- Modelo: Realiza la gestión de la información del sistema, encargándose del acceso, consulta y modificación de los datos. Las peticiones de información le llegan a través de los controladores.
- Controlador: Gestiona las peticiones o eventos que se producen en la interfaz de usuario, y se encarga de recoger, procesar y enviar los datos que se intercambian entre el modelo y la vista.
- Vista: Presenta la información del modelo de forma adecuada para poder gestionarla a través de, generalmente, una interfaz de usuario.

Las ventajas de utilizar un patrón como el MVC en el desarrollo de componentes software son múltiples y variadas:

- El desarrollo e implementación se realiza de forma modular, separando la gestión, el almacenamiento y la visualización de los datos en tres capas diferentes, lo que permite reducir al máximo el efecto que pueda producir cualquier cambio en el resto del sistema:
  - Las modificaciones que se realizan en la vista para modificar la interfaz no afectan a cómo se gestionan y almacenan los datos.
  - Las modificaciones que se realizan en el modelo de datos no afectan al resto de capas, salvo que estas modificaciones impliquen un cambio de alcance que requiera la ampliación o construcción de nuevas interfaces.
  - Cualquier cambio en la lógica del negocio es independiente de cómo se representan o almacenan los datos.
- Proporciona un sistema escalable y de fácil mantenimiento.
- Permite la construcción de sistemas distribuidos.
- Posibilita el desarrollo de múltiples interfaces de usuario sobre el mismo modelo de datos.

Aunque si bien es cierto que la aplicación del patrón MVC conlleva una mayor complejidad en el sistema desarrollado, fruto de la naturaleza modular del patrón, las ventajas que proporciona hace que sea muy habitual en la industria actual el desarrollo de software mediante patrones que implementan las características del patrón MVC. Ejemplos de patrones de programación MVC son Struts o Spring en Java, ASP.NET MVC en .NET o AngularJS en Javascript, sin duda algunos de los *frameworks* de programación más utilizados hoy en día.

5.3.1 Arquitectura SGBAP

La arquitectura de un SGBAP se ha desarrollado siguiendo el patrón MVC, como se ha comentado anteriormente, dividiendo la interfaz del usuario, la lógica de negocio y el modelo de datos en las tres vistas del patrón –modelo, vista y controlador-. En la siguiente imagen se presenta la arquitectura diseñada para un SGBAP:

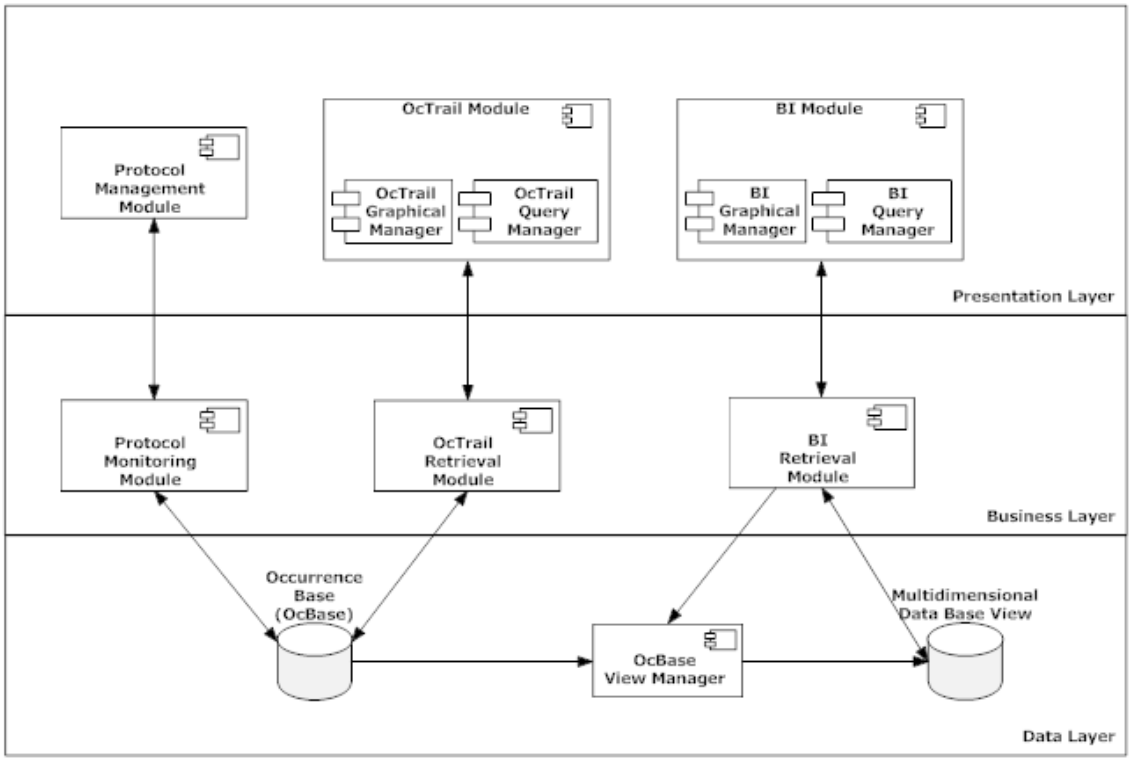


Figura 5-20 Arquitectura SGBAP –OcSystem-

Como se puede ver, las diferentes componentes que implementan la funcionalidad de un OcSystem están estructuradas entre las tres capas de este patrón: presentation layer –vista-, business layer –controlador- y data layer -modelo-.

La arquitectura desarrollada tiene como objetivo proporcionar tres funcionalidades diferentes para gestionar y explotar la información generada en los acontecimientos que se producen en las ejecuciones de protocolo. A continuación, se realiza primero una breve descripción de cada una de ellas y posteriormente las describiremos en detalle:

- **Gestión de Protocolos** (módulos de la rama izquierda de la arquitectura): En primer

lugar, se encuentra la propia implementación de los protocolos desarrollados para la ejecución de los procesos de negocio definidos por la organización para dirigir su trabajo de negocio diario. Estos módulos permiten al usuario registrar la ejecución de dichos protocolos, recogiendo los acontecimientos que se producen y almacenándolos en la OcBase.

- **OcTrail** (módulos de la rama central): Estos módulos permiten obtener un registro de auditoría o '*audit trail*' [8][80] de la información almacenada en el sistema en forma de acontecimientos. Un *audit trail* es un registro cronológico de las actividades o eventos que se realizan u ocurren en un sistema, de modo que puedan ser presentadas como evidencias de su actividad. Mediante el OcTrail se pueden extraer todos los acontecimientos que han ocurrido a un objeto y a todos sus objetos relacionados, permitiendo por tanto la obtención de las líneas de vida e historias de los objetos del sistema.
- **Business Intelligence** (módulos de la rama derecha): En la arquitectura se incluye también la implementación de un sistema de inteligencia de negocio (*Business Intelligence*) [35], mediante el que se podrá extraer conocimiento de los acontecimientos almacenados en la OcBase utilizando técnicas como los cubos OLAP para la obtención de indicadores y la realización de análisis de la información almacenada.

Para ilustrar estas funcionalidades, se mostrarán ejemplos de las interfaces de usuario desarrolladas en la herramienta ARASIS que veremos en detalle en el Capítulo 7.

5.3.2 Sistema de Gestión de protocolos

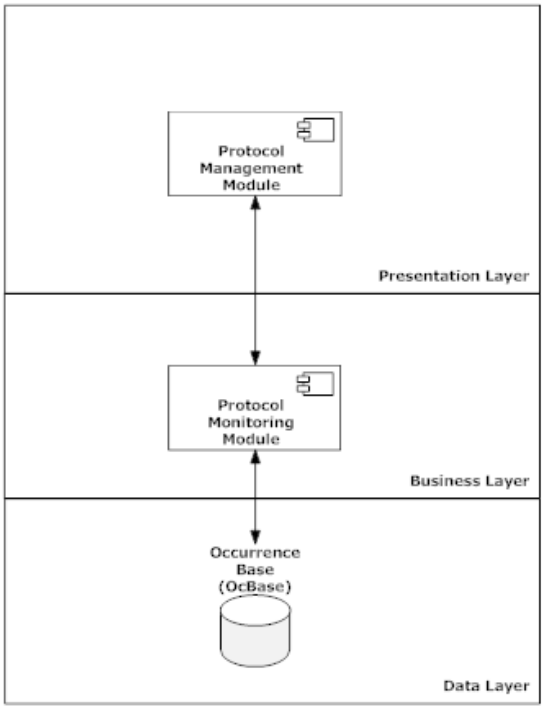


Figura 5-21 Arquitectura Sistema Gestión Protocolos

El sistema de Gestión de Protocolos, siguiendo la arquitectura MVC, está compuesto por tres componentes:

- **Protocol Management Module**. Proporciona la interfaz al usuario para gestionar la ejecución de los protocolos definidos en los procesos de negocio de la organización. Además de permitir la entrada de datos, proporciona la vista de la información almacenada en la OcBase del protocolo en ejecución y de todos los protocolos ejecutados a lo largo del tiempo.
- **Protocol Monitoring Module**. A medida que el usuario interactúa con el Protocol Management Module, el Protocol Monitoring Module recoge los acontecimientos que tienen lugar durante la ejecución de los protocolos y los almacena en la OcBase. Este módulo contiene toda la lógica de negocio de los protocolos, realizando las validaciones necesarias para asegurar la correcta ejecución de los mismos y obteniendo de la OcBase toda la información almacenada necesaria que solicite la interfaz de usuario.
- **OcBase**. Componente que representa la Base de Acontecimientos basados en Protocolos que estará desarrollada siguiendo la metodología especificada en la estrategia de diseño en el apartado anterior. Por tanto, contendrá las estructuras de persistencia que permiten el almacenamiento de los acontecimientos basados en protocolos, almacenando la información de los protocolos que se han ejecutado, la información de los objetos sobre los que se ejecutan, y los cambios de estado o de datos que han provocado. Estas estructuras estarán desarrolladas en función de los diagramas de estado que definen el comportamiento de los objetos, y que serán desarrollados como paso previo al desarrollo de los modelos PIM y PSM de la Base de Acontecimientos.

Un ejemplo de una interfaz de usuario para el sistema de Gestión de Protocolos se puede ver en la siguiente imagen:

Figura 5-22 Ejemplo de interfaz del Sistema de Gestión de Protocolos de ARASIS

En la imagen, se muestra la implementación realizada para la ejecución del protocolo de alta de alícuotas de la herramienta ARASIS, en la que se registra el proceso que se hace en el laboratorio de colocación de las alícuotas obtenidas de una muestra en sus objetos de almacenamiento, los racks.

En la siguiente imagen podemos ver otro ejemplo de interfaz del sistema de Gestión de Protocolos, en este caso, en el contexto de la gestión de incidencias según ITIL:

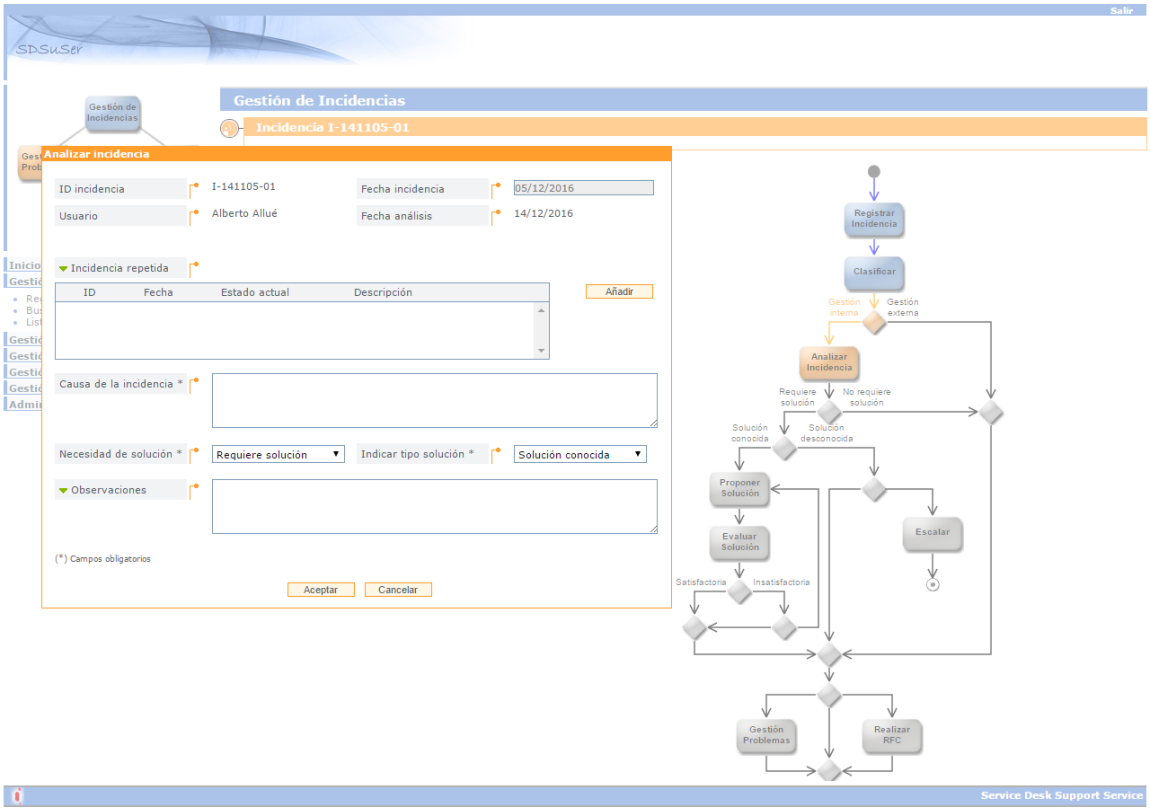


Figura 5-23 Ejemplo de interfaz del Sistema de Gestión de Protocolos en el contexto de ITIL

En el ejemplo de la imagen anterior, se visualiza una forma de ejecutar el protocolo de analizar un incidente según la buenas prácticas ITIL v3.

Como se puede observar en los dos ejemplos anteriores, el diseño de la interfaz puede adaptarse según las necesidades del contexto en el que se desarrolle el SGBAP. En el caso de la gestión de incidencias, se ha implementado de tal forma que se puede visualizar el diagrama de trabajo, o workflow, del proceso de negocio de gestionar un incidente en su conjunto, con todos los protocolos y las relaciones que existen entre ellos de forma visible para el usuario.

5.3.3 Sistema Octrail

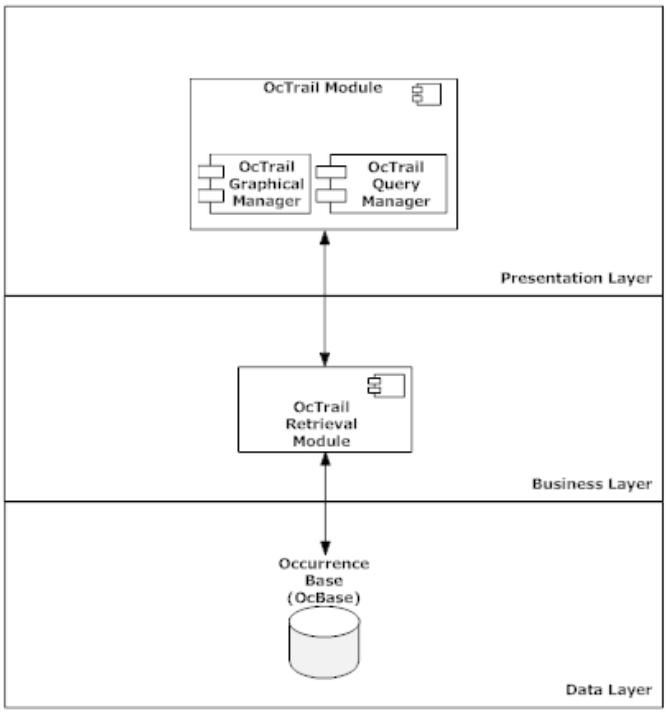


Figura 5-24 Arquitectura Sistema OcTrail

El sistema OcTrail, como se ha comentado en la introducción de la arquitectura, permite obtener los registros de auditoría, o '*audit trails*' en inglés, de los objetos del sistema. Los *audit trails* son utilizados en múltiples contextos debido a que muestran la actividad de un entorno de forma cronológica, lo cual es extremadamente útil para procesos como los de auditoría. Por ejemplo, se utilizan para registrar los accesos que se realizan a un sistema informático en el tiempo o para registrar las diferentes versiones de los elementos de un proyecto, como los planos de un edificio. Este registro no tiene que ser necesariamente informático o electrónico, sino que los *audit trails* pueden ser registros en papel que guarden una actividad de forma cronológica.

En el caso de la arquitectura, el concepto de *audit trail* se utiliza para obtener cronológicamente los acontecimientos que le han ocurrido a un objeto y a todos sus objetos relacionados. La representación de esta información se realiza con una interfaz de usuario que permite interaccionar con dichos acontecimientos para acceder a la información que forma parte de ellos: qué protocolo se ha ejecutado, sobre qué objeto y con qué efecto.

Por tanto, el sistema OcTrail nos proporciona los mecanismos necesarios para obtener los conceptos principales, tal y como se ha comentado en el apartado 3.3, que debe proporcionar cualquier Sistema Gestor de Bases de Acontecimientos: la historia y la línea de vida de los objetos. Para ofrecer esta funcionalidad, el sistema OcTrail se compone de tres componentes siguiendo la arquitectura MVC:

- **OcTrail Module**. Es la componente que gestiona la interfaz del sistema OcTrail de cara al usuario, y está formada por dos subcomponentes:
  - **OcTrail Query Manager**. Permite al usuario seleccionar y establecer condiciones

para consultar la información de la OcBase. Por ejemplo, el usuario podrá seleccionar un objeto concreto y establecer parámetros como el periodo de tiempo o el tipo de protocolo que marcarán el conjunto de acontecimientos que se obtendrán de la OcBase.

  - **OcTrail Graphical Manager**. Es la componente encargada de la representación gráfica de los conjuntos de acontecimientos resultantes según las condiciones establecidas en el Query Manager. En un SGBAP, esta representación gráfica se realizará longitudinalmente mediante la visualización en el tiempo de los acontecimientos. Esta visualización longitudinal de los acontecimientos permitirá consultar la historia de los objetos, mediante la representación de las líneas de vida de los objetos y de sus objetos relacionados. El usuario podrá acceder dinámicamente a la información de cada acontecimiento, avanzar en el tiempo para visualizar todos los acontecimientos del resto de la línea de vida, podrá acceder a la línea de vida de un objeto relacionado, etc.

- **OcTrail Retrieval Module**. Este módulo es el encargado de extraer la información de los acontecimientos almacenados en la OcBase para ser representada por el OcTrail Graphical Manager. Para hacer esto, este módulo transforma las condiciones que el usuario establece en el OcTrail Query Manager en un conjunto de consultas que serán formuladas contra la OcBase. Este módulo se aprovecha del diseño mediante los patrones de acontecimientos basados en protocolos con el que está construida la OcBase para que todas las consultas tengan la misma estructura, lo que le permite ganar en eficiencia. Para cada objeto, la consulta extrae los diferentes estados por los que ha pasado con la información del protocolo que provocó el cambio de estado y sus posibles datos asociados.
- **OcBase**. El sistema OcTrail utiliza para su funcionamiento los acontecimientos que se registran mediante el Sistema de Gestión de Protocolos en la Base de Acontecimientos basados en Protocolos OcBase.

En la siguiente imagen, se puede ver un ejemplo de la interfaz del sistema OcTrail desarrollado para la aplicación ARASIS de gestión de muestras biológicas.

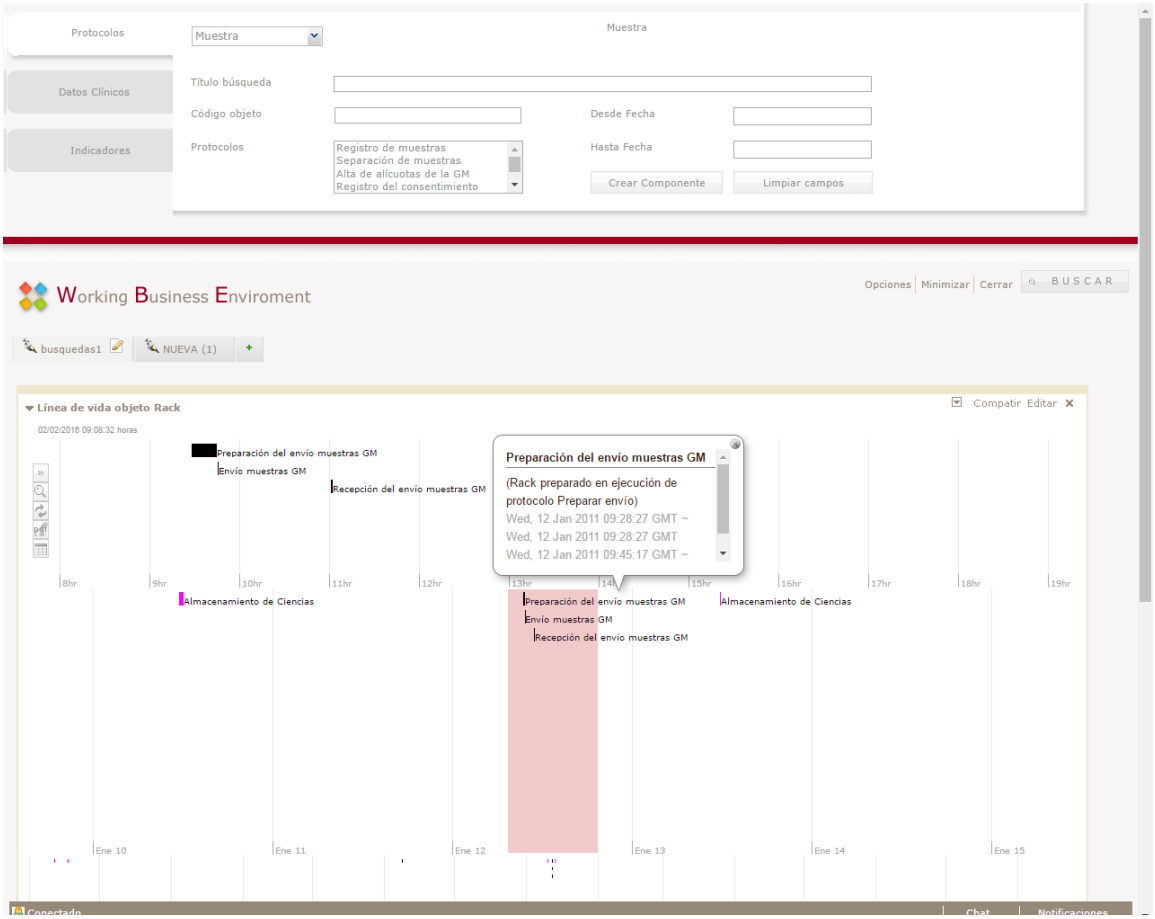


Figura 5-25 Ejemplo de interfaz del Sistema OcTrail

En la imagen anterior se puede apreciar la componente OcTrail Query Manager, en la parte superior, que permite realizar la selección inicial de las condiciones a buscar, y la componente OcTrail Graphical Manager, en la parte inferior, donde se muestra una línea de vida de los acontecimientos registrados para un objeto –con la información mostrada de un acontecimiento concreto- a lo largo del tiempo (un audit trail).

5.3.4 Sistema de Business Intelligence

La Inteligencia Empresarial [35] –Business Intelligence, BI, en inglés- se puede describir como el conjunto de técnicas utilizadas para el análisis de datos en la toma de decisiones de una empresa u organización. Estas técnicas pueden ir desde la confección de informes predefinidos que permitan extraer conocimiento básico de la información del sistema (por ejemplo, informes periódicos de balance económico), informes a medida según unas necesidades concretas (p. ej., la evolución de ventas de un artículo concreto) o complejos sistemas de análisis dinámico de la información como los sistemas OLAP [86][19].

En el caso de un SGBAP, el Sistema de Business Intelligence de la arquitectura utiliza la tecnología OLAP para extraer conocimiento de los acontecimientos registrados en la ejecución de los protocolos de los procesos de negocio.

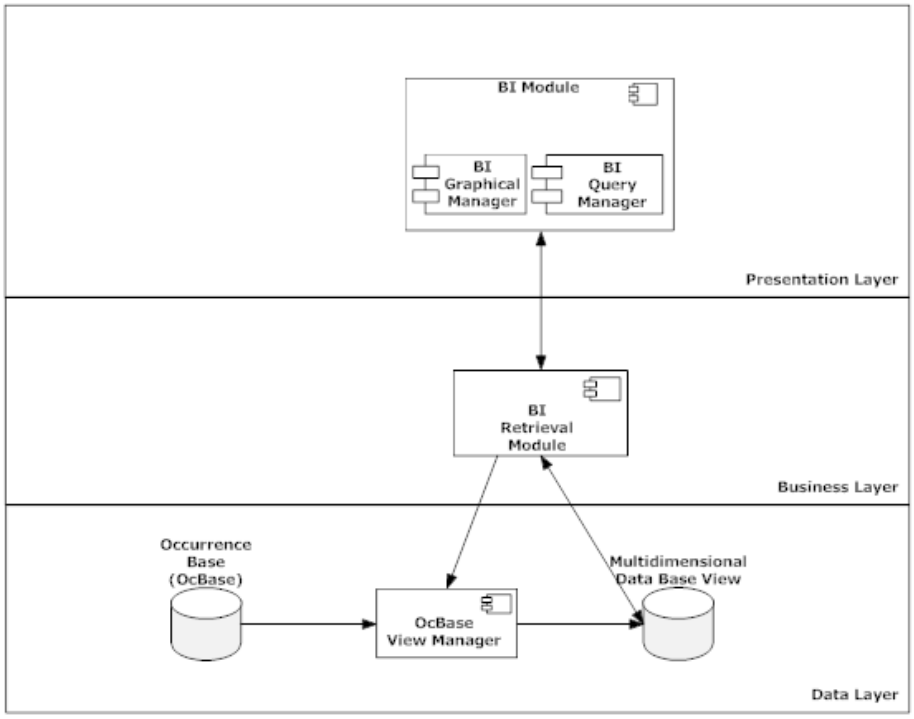


Figura 5-26 Arquitectura del Sistema Business Intelligence

Para implementar la tecnología OLAP en el SGBAP, el Sistema de Business Intelligence se compone, siguiendo el patrón MVC, de los siguientes componentes:

- **Business Intelligence Module**. Este módulo proporciona la interfaz al usuario que le permite extraer conocimiento relacionado con la ejecución de los procesos de negocio mediante cubos OLAP definidos en el sistema. Para ello, este módulo contiene dos subcomponentes:
  - **BI Query Manager**. Mediante esta componente el usuario puede seleccionar las opciones predefinidas que configurarán los cubos OLAP para obtener la información e iniciarán el proceso de análisis de datos: cubos, medidas, dimensiones, condiciones…
  - **BI Graphical Manager**. Se encarga de presentar la información de los cubos OLAP al usuario y de ofrecerle un entorno para que éste pueda interactuar dinámicamente, pudiendo filtrar la información obtenida, modificar la consulta al cubo OLAP…
- **BI Retrieval Module**. Es el encargado de implementar la tecnología OLAP contra la información de la OcBase. El BI Retrieval Module tiene construidos los posibles cubos OLAP de consulta que configurará en función de las selecciones que realiza el usuario en la interfaz, así como el motor de funcionamiento de la tecnología OLAP en dos vertientes:
  - Preparar y realizar las consultas MDX a los cubos OLAP.
  - Ejecutar las consultas del cubo OLAP sobre la OCBase.
- **Multidimensional Data Base View**. Como se ha comentado en los conceptos de OLAP, los cubos OLAP trabajan sobre esquemas de bases de datos multidimensionales (en estrella o en copo de nieve). Dado que la OcBase está construida como una base de datos relacional, el tipo de sistema OLAP utilizado en el SGBAP será un sistema ROLAP.

Para ello, en la capa de modelo se incluye la componente **OcBase View Manager** cuyo objetivo es la construcción de un sistema de vistas de bases de datos que ajustarán la estructura de la OcBase para los cubos OLAP, simulando una base de datos multidimensional utilizando un esquema en estrella (componente Multidimensional Data Base View).

En la siguiente imagen se puede ver un ejemplo de desarrollo de un Sistema de Business Intelligence de la arquitectura del SGBAP:

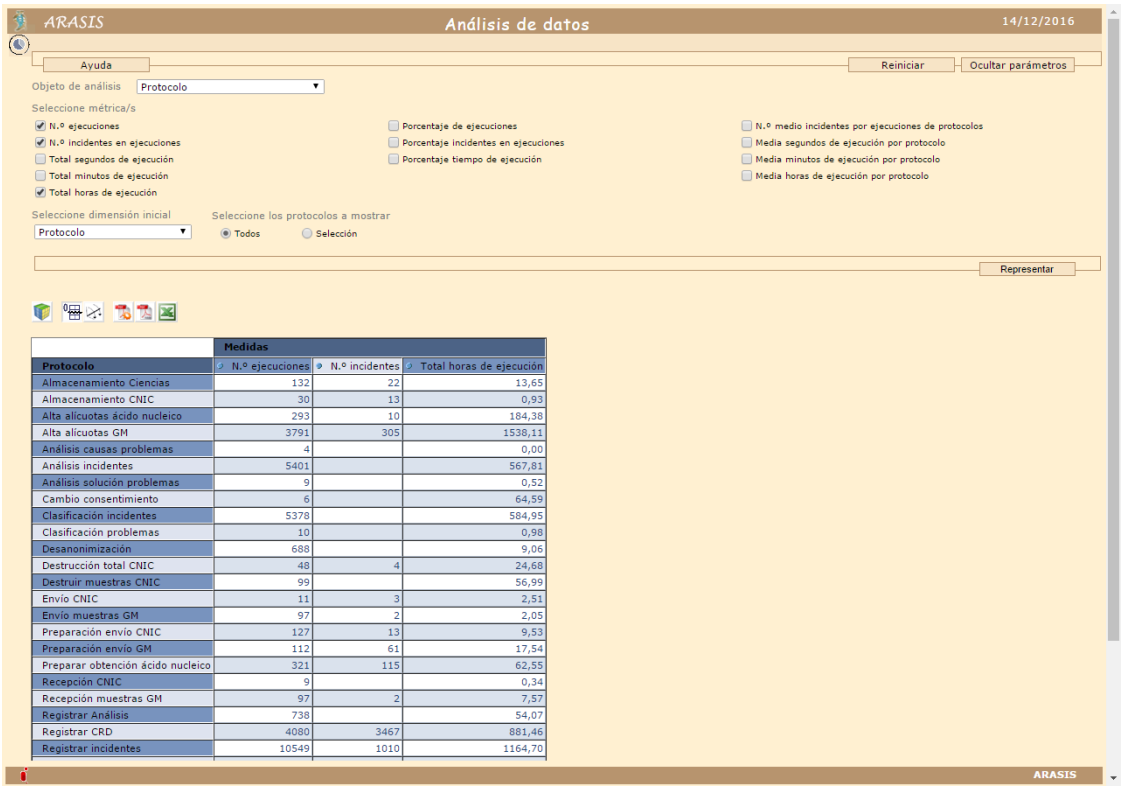


Figura 5-27 Ejemplo interfaz usuario del Sistema Business Intelligence

En la imagen anterior se puede apreciar la componente BI Query Manager, en la parte superior, que permite realizar la selección inicial de las dimensiones y métricas iniciales para la configuración de la consulta MDX inicial, y la componente BI Graphical Manager, en la parte inferior, donde se muestran los resultados obtenidos (en formato tabular) de la ejecución de las consultas MDX sobre los cubos OLAP diseñados, así como los iconos de acceso a las funciones dinámicas de interacción con el cubo.

Capítulo 6. Evolución de protocolos en los SGBAPs

Un problema crítico en los sistemas de monitorización de procesos es cómo responden a los continuos cambios en el negocio, ya que ello deriva en cambios en los procesos a los que se deben de adaptar estos sistemas. El problema habitual en este tipo de situaciones es que normalmente sólo se afronta la adaptación de los sistemas a nuevos flujos de trabajo que se produzcan en los procesos [24][29][30][40], pero no se analiza ni se resuelve cómo tratar también la información que ya estaba registrada [18][77].

Afrontar estas cuestiones es necesario para poder monitorizar y auditar la información recogida tanto en los modelos de procesos de negocio antiguos como en los nuevos, así como conocer los procesos que se han utilizado para almacenar ambos tipos de informaciones para saber si dicha información pertenece a un modelo o a otro. De esta forma, se pueden cumplir las propiedades de flexibilidad y consistencia que se especifican como propiedades de calidad de los modelos en [63].

En el caso de los SGBAPs, donde la información está organizada en base a un conjunto de protocolos que implementan los procesos de negocio, puede ser habitual que se necesiten cambiar los protocolos. Por esta razón, la arquitectura y metodología de la estrategia de diseño debe aportar una solución para afrontar dichos cambios. Por ejemplo, en el contexto de muestras biológicas, es posible que se definan nuevos protocolos de laboratorio para tratar las muestras que habrá que incorporar a los sistemas informáticos. También es posible que se detecte algún fallo en los propios protocolos de laboratorio que haya que corregir, o que no se haya trasladado correctamente el protocolo de laboratorio al sistema informático y sea necesario modificarlo para mantener la consistencia entre el mundo real y la información registrada en la base de acontecimientos [53][71].

La evolución de protocolos debido a estos motivos en los SGBAPs se debe considerar desde dos vertientes:

- En la base de acontecimientos. Se debe resolver cómo realizar la evolución de la base de acontecimientos desarrollada para el SGBAP a través de la metodología para que sea capaz de gestionar los cambios en los protocolos.
- En el conjunto de la arquitectura. La arquitectura del SGBAP debe permitir su adaptación a los cambios en los protocolos.

6.1 Adaptación de la base de acontecimientos

Los cambios en los protocolos de los procesos de negocio pueden conllevar cambios en la información almacenada en la base de acontecimientos, ya que estos cambios pueden suponer

nuevos protocolos, nuevos estados de los objetos, modificar los cambios de estados actuales o la información almacenada de los objetos.

En el contexto de las bases de datos, la evolución del esquema de datos es una cuestión ampliamente conocida, donde en general se proponen tres tipos de soluciones [4][75][16]:

- **Cambio del esquema,** *schema change*. Se sustituye completamente el esquema de datos actual por el resultante de la modificación.

  En este tipo de solución, el inconveniente es que los datos anteriores se descartan completamente, perdiendo cualquier tipo de información anterior al cambio de esquema desde el sistema actual.

  Esta solución se puede aplicar en situaciones en las que el esquema nuevo es radicalmente diferente al nuevo, y no se necesita mantener la información anterior en el sistema.

- **Evolución del esquema,** *schema evolution*. En este enfoque, el esquema actual se modifica para contemplar los cambios, añadiendo nuevos elementos y eliminando los que ya no se utilizan, con la consiguiente adaptación de la información ya registrada para que sea compatible con el nuevo esquema resultante.

  En esta solución, si bien no se pierde toda la información anterior registrada, sólo aquella que se elimina por los cambios, se pierde la referencia de cómo se ha registrado la información que sí se mantiene, desconociendo si es información registrada en función del anterior esquema o del nuevo.

  Este tipo de solución se puede aplicar en aquellas situaciones donde la evolución del esquema es mediante cambios sencillos que haga fácil la adaptación de la información ya registrada, y cuando no sea necesario preservar el cómo se ha registrado la información.

- **Versionado del esquema,** *schema versioning*. Este enfoque es el más complejo ya que, como su propio nombre indica, propone mantener un sistema de versionado de los esquemas de datos, de forma que pueda ser posible consultar la información registrada en cada versión según sea la situación.

  En la solución de versionado, no se pierde ninguna información registrada con anterioridad, ni cómo se han registrado, salvaguardando la procedencia de los datos -*data provenance*-.

  Esta solución es la más completa, y es la óptima cuando se necesita no perder información ni los esquemas de datos que se han utilizado para almacenarla.

  La implementación del versionado de esquemas se puede realizar de varias formas, siendo las dos más importantes las conocidas como '*pool único*' y '*pool múltiple*' de datos [16][41], refiriéndose al número de repositorios que se utilizan para almacenar las distintas versiones de esquemas de datos.

  En el *pool único* sólo existe un repositorio de datos, una base de datos, donde se almacenan todas las versiones del esquema de datos de manera conjunta, mientras que en el *pool múltiple* cada versión del esquema de datos se almacena en una base de datos diferente.

De forma análoga a estos conceptos de evolución de los esquemas en las bases de datos, se puede establecer un alineamiento con la evolución de los protocolos en el caso de bases de acontecimientos. Para ilustrar estos métodos seguiremos utilizando como ejemplo los casos de

ARASIS y de gestión de incidencias.

- **Cambio de protocolo**. Consistiría en cambiar completamente la estructura de persistencia que almacenaría la información de un protocolo en la base de acontecimientos, eliminando las estructuras e información anteriores.

  Al igual que en el caso de las bases de datos, este tipo de solución sirve para el caso en el que los protocolos que se sustituyen no requieren ser almacenados en la base de acontecimientos una vez realizada la modificación, incluyendo los acontecimientos que se generaron al ejecutarlos.

  Por ejemplo, imaginemos que se modifica la gestión de incidencias de un sistema para hacerla más completa. Inicialmente, sólo almacenaba la ejecución de un protocolo que realizaba el registro de un incidente y otro protocolo para realizar el cierre del mismo. La modificación consiste en cambiar el ciclo de vida del incidente por completo para adecuarlo a la guía de ITIL v3, incluyendo los estados de registro del incidente, su clasificación, análisis, etc. En este caso, podríamos utilizar el método de cambio de protocolo al cambiar de forma completa la gestión de los incidentes. Eso sí, la información de los incidentes gestionados por el sistema antiguo se eliminaría, almacenándola a efectos históricos en alguna estructura ad-hoc preparada para ello si fuera necesario preservarla.

- **Evolución de protocolo**. En este enfoque, el cambio consistiría en modificar la estructura de estados, cambios de estado o de información almacenada de los objetos para incluir los nuevos que provocan el cambio del protocolo.

  Por ejemplo, imaginemos en el sistema de gestión de incidentes anterior que en el protocolo de cierre del incidente únicamente se registraba el resultado de la resolución del incidente (si se ha solucionado o no). Sin embargo, el protocolo varía para incluir el registro de una valoración del usuario o cliente que ha reportado el incidente, ajustándose más fielmente a las buenas prácticas de ITIL v3.

  En este caso, se podría utilizar el método de evolución para incluir esta variación en el protocolo de dos formas: (1) definiendo estos datos como opcionales, quedando vacíos en las ejecuciones antiguas y pudiendo ser introducidos para las ejecuciones del nuevo protocolo, o, (2) dado que es información que se considera imprescindible, definirlo como obligatorio y realizar un proceso adicional para incluir un valor por defecto en las ejecuciones anteriores.

  Este tipo de solución, como se ve en el ejemplo, se puede aplicar para correcciones en los protocolos, o modificaciones, en aquellos casos en los que es posible dar alguna solución para las ejecuciones ya almacenadas de los protocolos antiguos.

- **Control de versiones de protocolo**. En el caso del control de versiones, al igual que en el caso de las bases de datos, la idea es mantener cada versión del protocolo existente en la base de acontecimientos, de forma que se pueda acceder a todos los datos. Dado que este enfoque es el más complejo, veámoslo con más detalle a continuación.

Al igual que en el caso de las bases de datos, para el versionado de protocolos podríamos optar por una solución de *pool único*, donde las diferentes versiones de los protocolos convivirían en la misma base de acontecimientos, o de *pool múltiple*, donde las versiones quedarían almacenadas en bases de acontecimientos diferentes.

En la estrategia de diseño que proponemos en esta tesis, la propuesta para implementar el

versionado de protocolos en los SGBAPs es utilizar el *pool único*, ya que permite una solución mucho más simple y económica en términos de recursos necesarios y complejidad de gestión que la solución de *pool múltiple*. Además, al tener únicamente una base de acontecimientos, también facilita todos los sistemas de copias de seguridad y recuperación ante errores que siempre deben de considerarse en la gestión de todo sistema informático.

Por otro lado, la utilización de esta opción permite que los sistemas de explotación de los acontecimientos (sistemas OcTrail y de Business Intelligence) no requieran de ninguna adaptación especial cada vez que se versione un protocolo, ya que a efectos de estos sistemas serán protocolos cualesquiera del sistema sobre los que recuperar acontecimientos que se hayan producido en el tiempo. En la información asociada a cada acontecimiento se indicará a qué versión del protocolo pertenece.

Por lo tanto, un SGBAP que implemente el versionado de protocolos permitirá no sólo generar y procesar los acontecimientos generados según la actualidad vigente de los protocolos, sino también acceder a los acontecimientos almacenados bajo cualquier versión histórica de los protocolos.

Como aspecto negativo de esta solución se encuentra, evidentemente, que al mantener la información tanto de los protocolos vigentes como de los históricos, el volumen de información almacenado irá creciendo a lo largo del tiempo, mermando paulatinamente la eficiencia del sistema.

Veamos el control de versiones en un SGBAP mediante un ejemplo. En concreto, la definición de un nuevo proceso de negocio en el envío de racks de alícuotas del caso ARASIS (utilizado en la explicación de la estrategia) entre diferentes instalaciones del biobanco. El nuevo proceso que se añade es la verificación de los racks de modo que, antes de enviar un rack, es necesario comprobar que las alícuotas incluidas en el rack son las correctas. La definición de este nuevo proceso, además de desarrollar un nuevo protocolo que lo implementa, provoca la modificación del protocolo de envío de un rack que ya existe actualmente en el sistema puesto que sólo se pueden enviar racks que hayan sido verificados previamente. En la siguiente imagen puede verse la situación de partida que se tiene en el sistema:

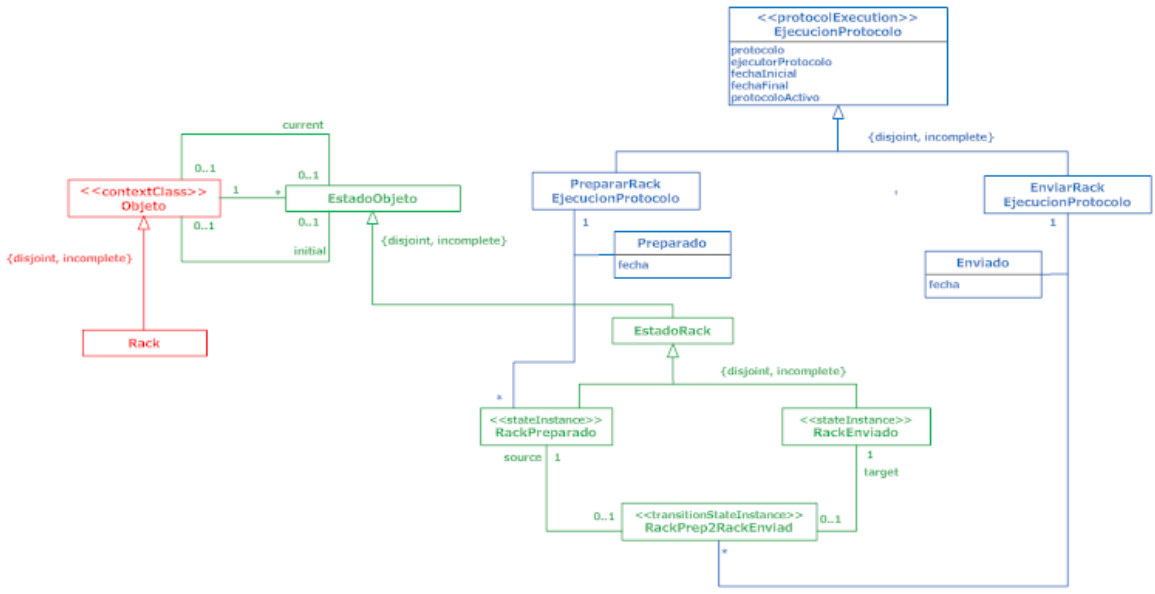


Figura 6-1 Ejemplo de evolución de protocolos en la OcBase en el modelo PIM –Estado inicial

Como se puede ver, inicialmente tenemos dos estados, preparado y enviado, con el protocolo de envío original que provoca la transición entre ellos.

Para incluir el nuevo proceso en la base de acontecimientos, como se ha comentado, se necesitará definir un nuevo protocolo, el protocolo de 'Verificar envío'. Además, como consecuencia de la aplicación de este protocolo a los racks, en la base de acontecimientos es necesario definir un nuevo estado al objeto 'rack' que llamaremos 'Verificado'. El efecto del nuevo protocolo será que el rack pasará del estado 'preparado' a 'verificado'.

Tal como hemos indicado, como consecuencia de la inclusión del nuevo proceso, el protocolo 'Enviar rack' será modificado de modo que sólo pueden enviarse aquellos racks que hayan sido verificados. De esta forma, el efecto del protocolo de envío ya no es que el rack en estado 'preparado' cambia al estado 'enviado', sino que el cambio de estado provocado por el protocolo es de 'verificado' a 'enviado'.

En la siguiente imagen puede verse la aplicación de esta modificación siguiendo la propuesta de *pool único* para el control de versiones:

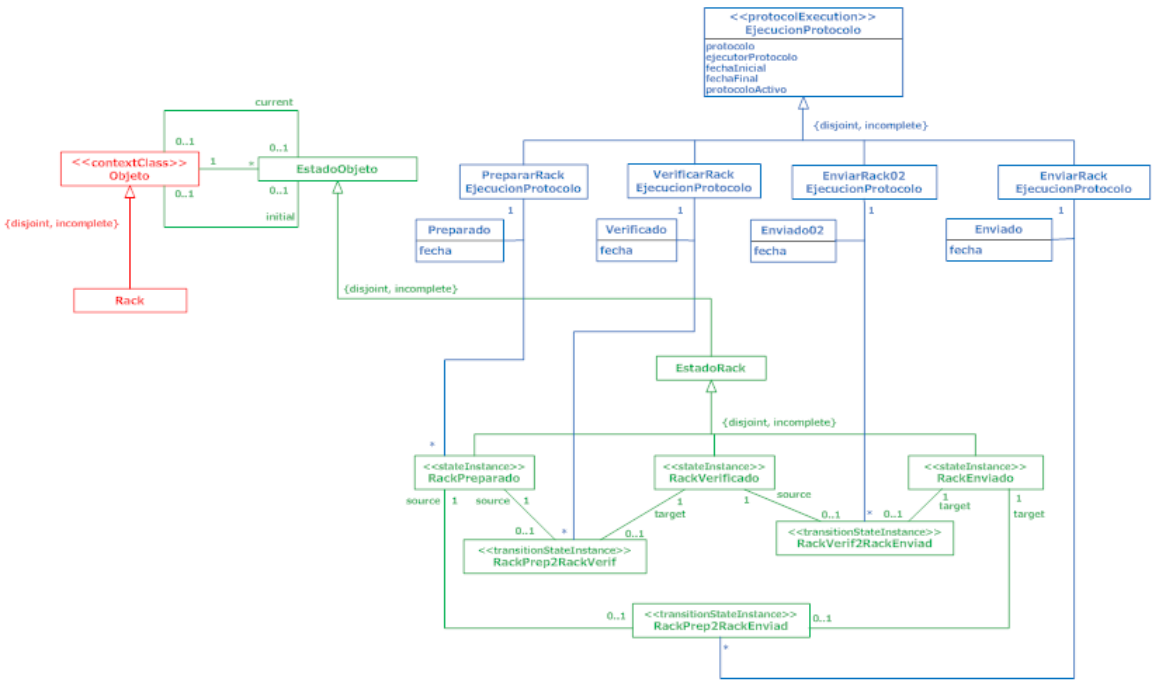


Figura 6-2 Ejemplo de evolución de protocolos en la OcBase en el modelo PIM –Estado final

Como se puede ver, se añade el nuevo protocolo de 'VerificarEnvio' y la nueva versión del protocolo de envío, el protocolo 'EnviarRack02'. El efecto del protocolo de verificación, como se ha comentado, es el cambio de estado de 'RackPreparado' a 'RackVerificado' mediante la transición 'RackPrep2RackVerif', mientras que el del nuevo protocolo de envío, es el cambio de estado de 'RackVerificado' a 'RackEnviado' mediante la transición 'RackVerif2RackEnviad'.

Respecto al protocolo de envío antiguo, el protocolo 'EnviarRack', se mantiene en la estructura de la base de acontecimientos con la transición original 'RackPrep2RackEnviad' del estado

'RackPreparado' al estado 'RackEnviado', lo que nos permite mantener la información registrada a través de este protocolo en la base de acontecimientos, además de conocer en todo momento si la información de un rack ha sido registrada mediante el protocolo de envío antiguo o el nuevo, conservándose, por tanto, el *provenance* de los datos.

Ahora bien, queda pendiente el cómo distinguir en el sistema para los protocolos de envío cuál es el protocolo correcto y cuál no. Para ello, haremos uso del atributo 'protocoloActivo' que se definió en el perfil de acontecimientos, y que estaba incluido en los patrones PIM y PSM, de la Base de Acontecimientos –apartado 5.1-.

En este ejemplo, en el caso del protocolo 'EnviarRack' se pondrá a 'false', indicando que es un protocolo que no está activo en el sistema, mientras que en el protocolo 'EnviarRack02' este valor estará a 'true' (este hecho no se incluye en la figura anterior ya que es un valor de instancia en tiempo de ejecución del sistema).

Respecto a cuál es la mejor solución a aplicar ante un cambio de protocolo dependerá de la naturaleza del cambio y, sobre todo, del entorno en donde se haya construido el SGBAP. La solución de control de versiones de protocolos realmente puede aplicarse en cualquier contexto, siendo la solución necesaria para aquellos casos en los que sea requisito el poder conocer en cada momento qué información estaba registrada y en base a qué protocolo. Es el caso, por ejemplo, de sistemas para la gestión de muestras biológicas o de entornos sensibles en cuanto a la información almacenada, donde es indispensable el almacenamiento de cualquier dato durante una cantidad de años ante posibles solicitudes judiciales, o en sistemas para auditorías de la información. En estos casos, la solución de control de versiones proporciona los mecanismos necesarios para poder conocer, en cualquier momento, con qué protocolo fueron almacenados los datos.

Sin embargo, las otras opciones, el cambio y la evolución de protocolo, también nos proporciona una solución menos compleja para aquellas situaciones en las que no se necesita almacenar información histórica o, al menos, no manteniendo la información exacta de con qué protocolos se han registrado.

6.2 Adaptación de la arquitectura

Una vez que tenemos una solución para la adaptación de la Base de Acontecimientos a cualquier cambio que surja sobre los protocolos, lo siguiente es saber cómo se comportaría el resto de elementos de la arquitectura ante el cambio. En este caso, gracias a la arquitectura desarrollada mediante el patrón MVC los cambios son mínimos, demostrando la escalabilidad y mantenimiento de la arquitectura propuesta.

Los cambios a realizar en los diferentes sistemas de la arquitectura serían:

- Sistema de Gestión de Protocolos. En este sistema los cambios a realizar dependerán de si hay que considerar un nuevo protocolo, donde habrá que añadir lo necesario en las tres capas de la arquitectura, o es la modificación de uno existente, donde dependiendo del cambio, es posible que sólo haya que modificar la conexión del controlador con el modelo para gestionar el cambio de datos, salvo que la modificación también requiera de la adaptación de la interfaz de usuario.

  En el caso del ejemplo de control de versiones del apartado anterior, con el nuevo

proceso de negocio de verificación de pedidos, tenemos reflejadas ambas situaciones.

La inclusión del nuevo protocolo de 'Verificar envío' implicaría desarrollar la interfaz correspondiente en el módulo 'Protocol Management Module' y la lógica de negocio necesaria en la componente 'Protocol Monitoring Module' para su gestión con la Base de Acontecimientos (modificada siguiendo la imagen del apartado anterior).

La modificación del protocolo de envío, sin embargo, sólo significaría modificar los métodos de gestión con la base de datos, para obtener de cara a la interfaz del protocolo sólo aquellos pedidos que estén en estado 'Verificado', y modificar las conexiones con la base de datos para tratar el nuevo esquema de cambios de estado, registrando el cambio desde el estado de verificado en vez de preparado.

- Sistema de OcTrail. En el sistema de OcTrail, los cambios a realizar únicamente afectarían a la componente 'OcTrail Retrieval Module' para modificar la obtención de los acontecimientos en función de los posibles cambios de protocolos, ya que esta componente proporciona la información a las componentes de la interfaz en un formato definido independiente de la información –acontecimientos- que se va a representar.

  Siguiendo con el ejemplo de control de versiones del apartado anterior, en este caso sólo habría que añadir el tratamiento de los nuevos protocolos existentes (el de verificación de envío y el nuevo de enviar).

- Sistema de Business Intelligence. En el caso del sistema de Business Intelligence, los cambios afectarían a dos capas: la capa controlador y la capa del modelo.

  En el caso de la capa del modelo, los posibles cambios a realizar serían la adaptación de la componente 'OcBase View Manager' para contemplar nuevos protocolos, estados o transiciones, si existen, en el sistema multidimensional de vistas para los Cubos OLAP definidos.

  En el caso de la capa de controlador, al igual que en el sistema OcTrail, los cambios en la componente 'BI Retrieval Module' sería para adaptar la definición de los cubos OLAP a los nuevos protocolos definidos –en concreto, en la tabla de hechos-.

  En este sistema, como también ocurre en el sistema OCTrail, las componentes de la interfaz trabajan de forma dinámica con la información que proporcionan los cubos OLAP, por lo que no sería necesaria su modificación.

Capítulo 7. Implementación de un SGBAP: el caso ARASIS

Como se ha comentado en el capítulo introductorio de la tesis, como prueba de concepto de la estrategia de diseño se ha construido la herramienta ARASIS. La construcción de ARASIS se ha enmarcado dentro del proyecto AWHS [15], en el que se necesitaba la construcción de la herramienta informática que permitiera gestionar un biobanco de muestras biológicas. Para hacer toda esta gestión, se desarrollaron varios protocolos de laboratorio que definen las actividades que han de realizarse para la correcta gestión de las muestras biológicas, desde su obtención y almacenamiento en el biobanco, hasta su posterior tratamiento. ARASIS es la herramienta informática que se ha desarrollado para registrar toda la información de la ejecución de dichos protocolos de laboratorio, a partir de su implementación como protocolos informáticos con el registro de los datos de las actividades que realizan físicamente en los laboratorios.

ARASIS se ha desarrollado como un SGBAP, aplicando los conceptos de acontecimientos basados en protocolos y la estrategia de diseño que se han presentado en el Capítulo 5 de la tesis. Por otra parte, como sistema gestor de bases de acontecimientos, también proporciona los mecanismos que permiten explotar todo ese conocimiento para realizar los análisis de datos que se requieren en el estudio.

En este capítulo se detalla cómo se ha aplicado esta estrategia en la construcción de ARASIS, describiendo también tecnologías externas que se han utilizado para ello, y aprovechando esta aplicación para realizar un análisis y evaluación de la utilización de los acontecimientos basados en protocolos en un sistema real y exigente como ARASIS.

7.1 Aplicación de la estrategia de diseño en ARASIS

Para aplicar la estrategia de diseño definida en el Capítulo 5 para la construcción de ARASIS, hay que seguir los dos pasos que se indicaban en dicho capítulo:

- Desarrollo de la base de acontecimientos.

  Para ello, se deben definir los modelos PIM y PSM, que nos van a permitir construir la base de datos relacional en la que se va a implementar la base de acontecimientos. Para la realización de este paso necesitaremos establecer previamente los diferentes objetos, protocolos y diagramas de estado que nos permitirán definir dichos modelos con los elementos concretos del sistema ARASIS.

  Si recordamos el concepto general de base de acontecimientos, una de las características que se podían establecer era si se podían eliminar o no acontecimientos

registrados en el tiempo. En el caso de ARASIS, al ser una herramienta para la gestión de biobancos en los que hay un margen legal de tiempo en el que se puede requerir información de las muestras almacenadas, se estableció como requisito que no pudiera eliminarse la información generada en el transcurso del tiempo. Por este motivo, en aquellos objetos en los que es posible que se eliminen en el mundo real el elemento al que representan, como las muestras, se diseñaron estados finales en los que se registra el acontecimiento de que el objeto ha sido eliminando, manteniendo todos los acontecimientos registrados hasta entonces.

- Desarrollo de los mecanismos definidos en la arquitectura para almacenar y explotar el conocimiento de la base de acontecimientos. Es decir, desarrollar en ARASIS los sistemas definidos en la arquitectura de un SGBAP: Sistema Gestión de Protocolos, Sistema OcTrail y Sistema de Business Intelligence.

7.1.1 Base de acontecimientos de ARASIS

Para construir la base de acontecimientos basados en protocolos de ARASIS, el primer paso es definir los objetos en el sistema, sus diagramas de estado y los protocolos que van a actuar sobre ellos, los cuales van a provocar los cambios de estado de los objetos –acontecimiento basado en protocolos-.

En el caso de ARASIS, cuyo objetivo es recoger informáticamente la gestión de un biobanco, existen 9 objetos diferentes:

- Empleado: son los sujetos de estudio del proyecto AWHS, los empleados de la GM.
- Muestra: muestras biológicas sin analizar extraídas de los empleados.
- Alícuota: son las muestras divididas para el estudio –alicuotadas- en recipientes que se llaman alícuotas.
- Gradilla: es el contenedor donde se almacenan los tubos de muestras.
- Cubilete: es el contenedor donde se almacenan las muestras que son de tipo ADN.
- Rack: es el contenedor donde se almacenan las alícuotas.
- Jaula: es el contenedor donde se almacenan los racks.
- Congelador: es el contenedor donde se almacenan las jaulas.
- Placa: son los contenedores utilizados para realizar la cesión de las muestras a otros laboratorios.

Sobre los 9 objetos anteriores se ejecutan, actualmente, 66 protocolos diferentes que guían el comportamiento en el sistema de dichos objetos a través de sus estados. En la ejecución de los protocolos, además, pueden originarse incidentes que pueden ser gestionados a través de un pequeño sistema de gestión de incidentes según ITIL integrado en ARASIS.

Para dar una estimación aproximada del volumen de información que se gestiona en ARASIS, en la siguiente tabla se muestra la cantidad de datos que existe actualmente (a fecha de diciembre de 2016) en la base de acontecimientos según el tipo de objeto o de elemento gestionado:

Elemento	Cantidad
Empleados	6600
Muestras	77.541
Alícuotas	219.372
Gradilla	239
Cubilete	1.290
Rack	2.764
Jaula	146
Congelador	18
Placa	266
Protocolos	66
Ejecuciones de protocolo	70.316
Incidentes	15.274
Número de objetos afectados por incidentes	1.519

Figura 7-1 Volumen de información en ARASIS

Con el objetivo de explicar el proceso de aplicación de la metodología de construcción de la base de acontecimientos basados en protocolos seguida para ARASIS, vamos a utilizar únicamente dos tipos de objetos: el empleado y la muestra. Para el resto de tipos de objetos, el proceso que se explica a continuación será equivalente, pero considerando en cada caso los diagramas de estado y protocolos particulares de cada tipo de objeto.

En el caso del objeto empleado, el diagrama de estados que define su comportamiento en el sistema ARASIS puede verse en la siguiente imagen:

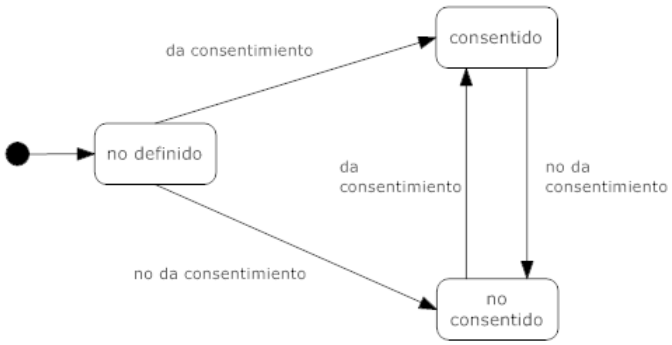


Figura 7-2 Diagrama de estados del consentimiento de un empleado

Los posibles estados de un empleado hacen referencia a si ha dado su consentimiento o no para que las muestras que le hayan sido extraídas puedan ser utilizadas en el estudio que se

realiza en el proyecto AWHS. Cuando se extraen las muestras a un empleado, se registra en ARASIS, e inicialmente tiene el estado 'no definido' puesto que se desconoce aún si va a consentir o no la utilización de sus muestras.

Con el motivo de no parar todo el proceso hasta el momento en que el empleado da respuesta al consentimiento, el departamento médico va trabajando con las muestras obtenidas, de tal forma que si posteriormente el empleado no da el consentimiento, las muestras correspondientes son destruidas (estado 'destruida' de las muestras).

Pasados unos días desde la extracción de las muestras se cita al empleado en la consulta médica, y es en ese momento cuando el empleado responde si da el consentimiento o no del material extraído.

Una vez dado el consentimiento, el empleado, en cualquier momento, puede dejar de dar el consentimiento a las muestras, lo que provocará que a esas muestras no se les pueda aplicar ningún proceso más, quedando en el estado en el que estuvieran.

Los protocolos que se aplican en este comportamiento del objeto empleado son:

- Protocolo 'Registrar muestras'. Durante el protocolo de registrar las muestras, es cuando se registra al empleado con el estado 'no definido', además de las muestras que se le extraen como veremos en el diagrama de estados del objeto muestra. Este protocolo es el que creará la instancia del objeto empleado en el sistema.
- Protocolo 'Registrar consentimiento'. Protocolo que se encarga de registrar el consentimiento o no consentimiento del empleado, provocando las transiciones del estado 'no definido' a los estados 'consentido' o 'no consentido' según sea el caso.
- Protocolo 'Cambiar consentimiento'. Protocolo que se encarga de recoger el cambio de consentimiento del empleado, activando las transiciones entre los estados 'consentido' y 'no consentido' según sea el estado en el que se encontrara el empleado.

El diagrama de estados que define el comportamiento en el sistema ARASIS en el caso del objeto muestra puede verse en la siguiente imagen:

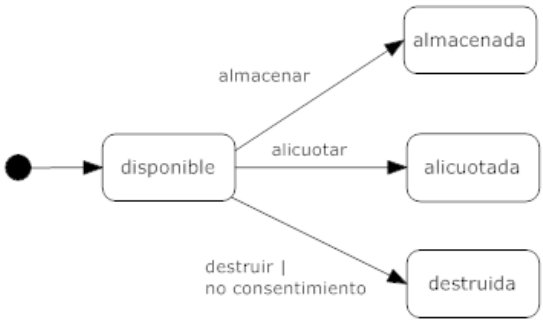


Figura 7-3 Diagrama de estados del objeto muestra

Para el objeto muestra, sus posibles estados son cuatro: (1) ha sido extraída y está disponible para el estudio, (2) ha sido almacenada para ser guardada en el biobanco para un futuro, (3) ha sido ya alicuotada para ser estudiada en el laboratorio y (4) ha sido destruida del biobanco debido a procesos de gestión del biobanco o por el no consentimiento del sujeto de estudio a

que sus muestras se utilicen en el proyecto AWHS.

Los protocolos por tanto que guían el comportamiento del objeto muestra son:

- Protocolo 'Registrar muestras'. Protocolo con el que se realiza el registro de las muestras. Este protocolo es el que creará la instancia del objeto muestra en el sistema, en el estado 'disponible'.
- Protocolo 'Registrar consentimiento'. En el protocolo que se registra el consentimiento del empleado, en caso de que se registre como 'no consentido', se eliminan sus muestras, provocando las transiciones del estado 'disponible' al estado 'destruida'. El motivo de este estado en la muestra es, como se ha comentado anteriormente, debido a que en el SGBAP de ARASIS se ha determinado que no puede eliminarse ningún acontecimiento. De esta forma, todos los acontecimientos registrados sobre las muestras permanecen en el sistema.
- Protocolo 'Eliminar muestras'. Las muestras pueden ser eliminadas no sólo al registrar el no consentimiento del empleado, sino por otros motivos del laboratorio. Este protocolo realiza este proceso, provocando la transición del estado 'disponible' al estado 'destruida'.
- Protocolo 'Alicuotar muestra'. Protocolo por el cual una muestra se alícuota para su tratamiento, provocando la transición del estado 'disponible' al estado 'alicuotada'.
- Protocolo 'Almacenar muestra'. Protocolo en el que la muestra es almacenada en una gradilla, disparando la transición del estado 'disponible' al estado 'almacenada'.

Una vez analizados los objetos, identificando sus posibles estados y cambios de estado debido a sus protocolos, el primer paso de la metodología para la construcción de la base de acontecimientos basados en protocolos es definir el modelo PIM independiente de la plataforma. Este modelo, si recordamos, se construía a partir de la repetición a modo de bloques de construcción de los patrones de acontecimientos.

En el caso de los empleados y las muestras, la aplicación del Patrón de Cambio de Estado da como resultado los fragmentos de modelos que se pueden ver en las siguientes imágenes:

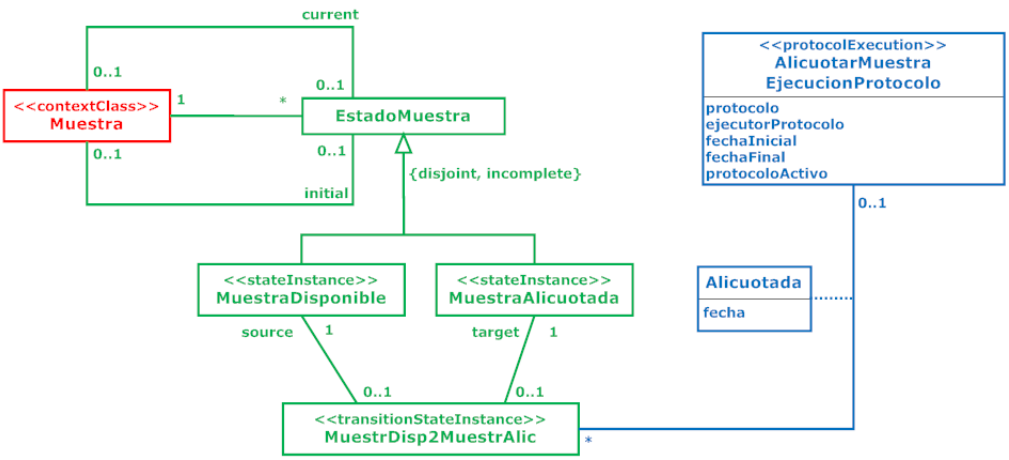


Figura 7-4 Patrón de Acontecimientos de Cambios de Estado en ARASIS para el objeto muestra

Como ejemplo de aplicación del patrón, se ha utilizado el cambio del estado 'disponible' al estado 'alicuotada' a través del protocolo 'AlicuotarMuestra'

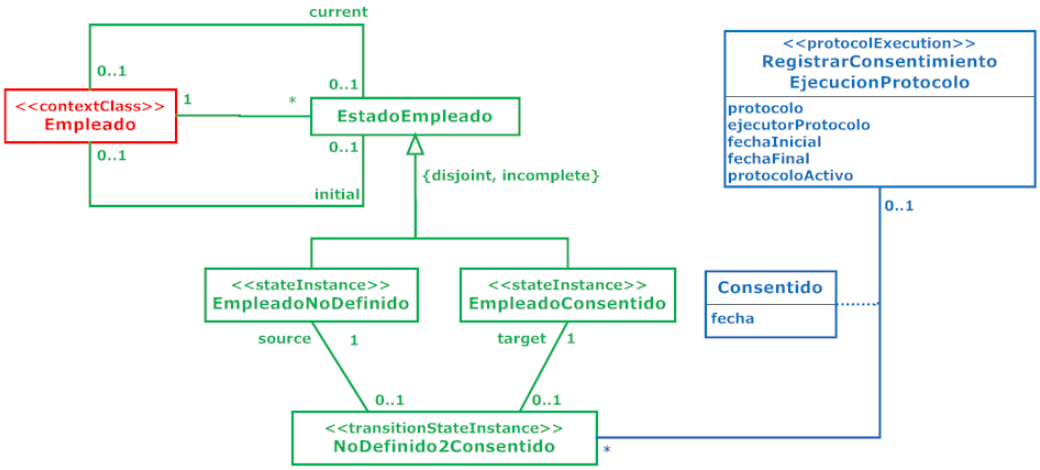


Figura 7-5 Patrón de Acontecimientos de Cambios de Estado en ARASIS para el objeto empleado

En el caso de la aplicación del patrón para el objeto empleado, se ha utilizado el cambio del estado 'no definido' al estado 'consentido' mediante el protocolo 'RegistrarConsentimiento'.

En el caso del Patrón de Creación del Objeto, el diseño para ambos objetos quedaría como se puede ver en las siguientes imágenes:

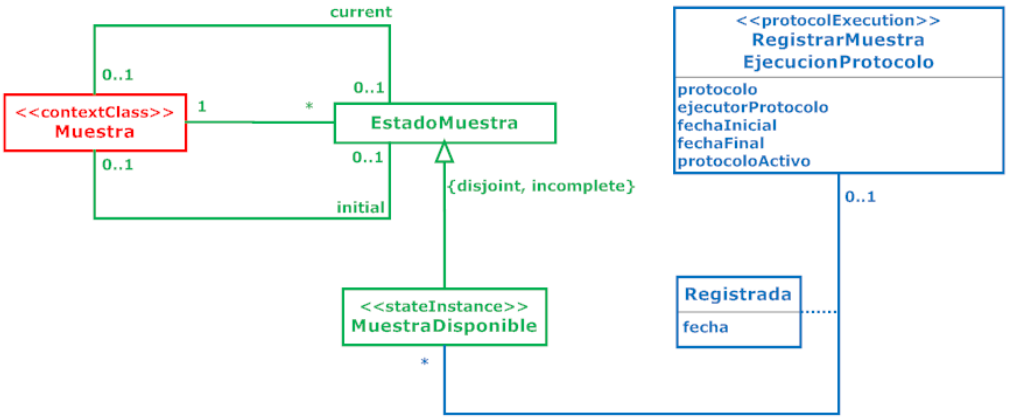


Figura 7-6 Patrón de Creación de Objeto en ARASIS para el objeto muestra

La aplicación del patrón se realiza utilizando el protocolo 'RegistrarMuestra', el cual crea el objeto en el estado 'disponible'.

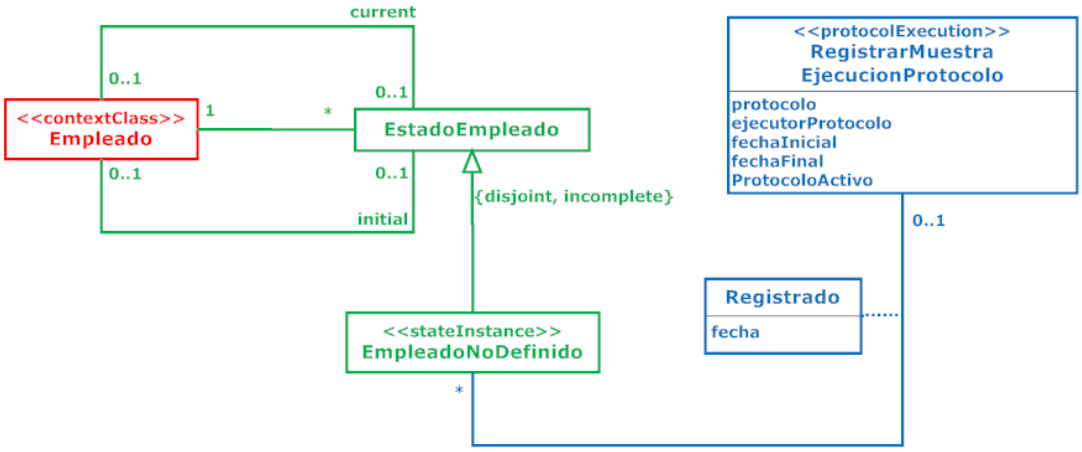


Figura 7-7 Patrón de Creación de Objeto en ARASIS para el objeto empleado

En el objeto empleado, la aplicación del patrón también se define con el protocolo 'RegistrarMuestra', en el cual se registra el empleado en el estado 'no definido'.

Por otro lado, en ARASIS también se utilizan patrones de datos para el registro de datos longitudinales. En concreto, se utilizan el Patrón de Registro de Datos de Objeto y el Patrón de Cambio de Datos de Objeto para realizar el registro longitudinal de la información de:

- Análisis clínicos de sangre y orina.
- Cuadernos de Recogida de Datos, llamados CRD, donde se registra información del peso, talla, tensiones, etc. que se le miden al empleado en las revisiones.
- Formularios de salud, donde se recogen las respuestas del empleado a preguntas sobre hábitos de consumo y de salud.

A continuación, se muestra la aplicación de estos patrones al caso de los CRD:

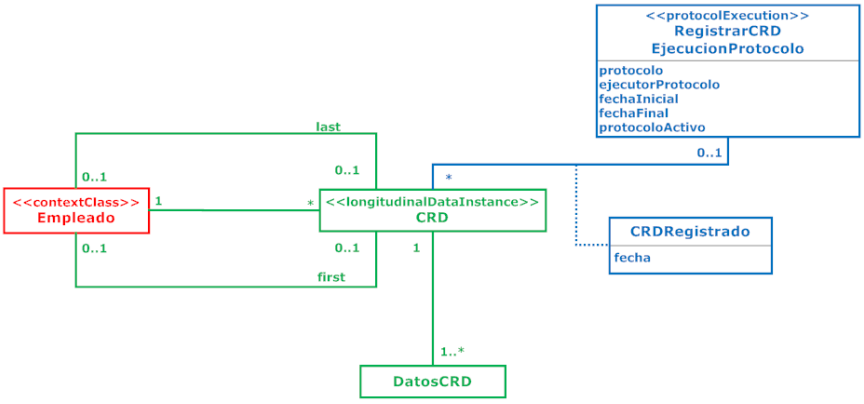


Figura 7-8 Patrón de Registro de Datos de Objeto en ARASIS

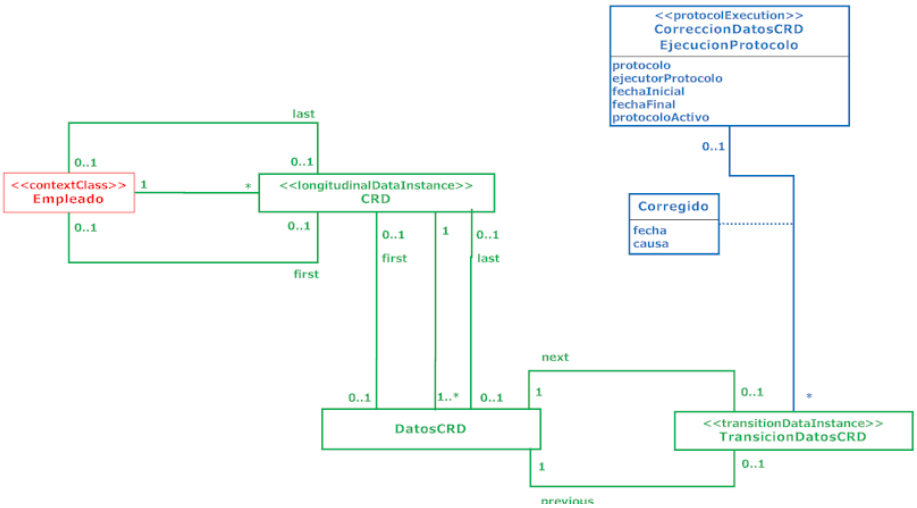


Figura 7-9 Patrón de Cambio de Datos de Objeto en ARASIS

Una vez aplicados los patrones de acontecimientos, el siguiente paso es desarrollar el modelo PIM del sistema, en el que se especificará el alcance del mismo. El objetivo de construir el modelo PIM, como se comentó en la estrategia, es tener definido el alcance del sistema de forma independiente a su plataforma de implementación, lo que nos facilitará realizar el proceso de análisis, evolución y mejora del sistema.

El modelo PIM de la base de acontecimientos, como se explicó en la estrategia, se construye repitiendo estos patrones de diseño para todos los objetos, estados y protocolos que existen en el sistema. Por ello, en la metodología se definía el patrón de diseño del modelo PIM –el patrón PIM-.

El modelo PIM de ARASIS se desarrolla utilizando el patrón PIM (ver Figura 5-13 del apartado 5.2) como base para la aplicación de los patrones de acontecimientos, representando los objetos, protocolos, estados y datos longitudinales que definen el alcance del sistema. En la siguiente imagen puede verse un extracto del modelo PIM que se construyó para la base de acontecimientos de ARASIS, acotando el alcance del diseño a los dos objetos con los que estamos trabajando, el empleado y la muestra.

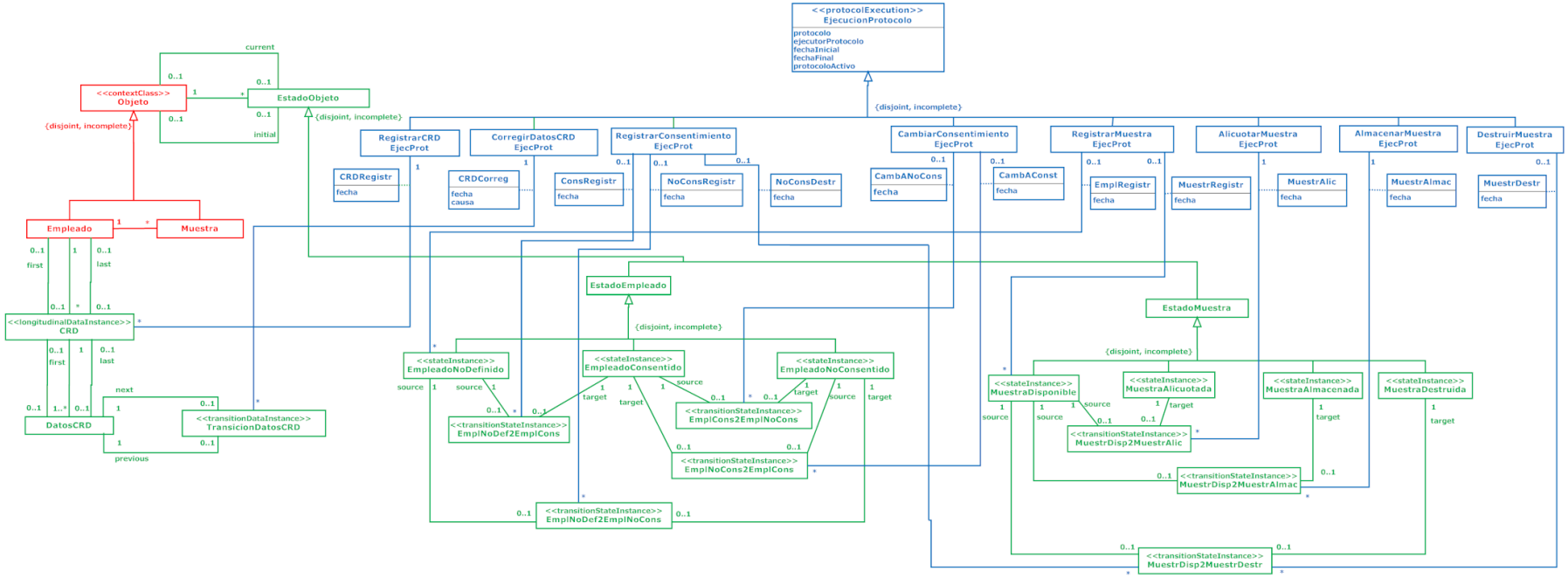


Figura 7-10 Extracto modelo PIM en ARASIS

Si analizamos en detalle el extracto del modelo PIM de ARASIS anterior, veremos claramente cómo se han aplicado los patrones de acontecimientos, y cómo queda perfectamente reflejado el comportamiento de los objetos en el sistema.

A la izquierda, en rojo, están definidos los objetos del sistema –empleado y muestra en el ejemplo-, con las relaciones que existen entre ellos, que en este caso es que un empleado puede tener varias muestras y cada muestra pertenece a un empleado. En el modelo se han obviado por claridad datos estáticos de los objetos, como el código que identifica al empleado en el sistema.

En la zona superior, en azul, están todos los protocolos que actúan en el sistema y que generan la información de los acontecimientos. Como vemos en el extracto, están definidos todos los protocolos que hemos detallado en el análisis inicial de los objetos.

En la parte inferior derecha, en verde, están los estados de los objetos. En el lado izquierdo están los estados y transiciones de estado para el objeto empleado, y en el lado derecho para el objeto muestra. Como puede verse, queda reflejado perfectamente el diagrama de estados de cada objeto, por lo que podemos en todo momento conocer no sólo el alcance de la información que va a ser registrada, sino también el comportamiento de un objeto en el sistema. Esto nos proporciona seguridad y calidad en la implementación, ya que se definen en el modelo de datos de forma intrínseca las posibles relaciones existentes, lo que va a evitar errores en el registro de la información.

Se incluyen todas las particularidades que se definían en el análisis inicial de los objetos:

- El protocolo Registra Muestra está asociado con los estados iniciales tanto del objeto muestra como del objeto empleado, ya que es el protocolo de creación de los objetos.
- Los protocolos Registrar Consentimiento y Cambiar Consentimiento pueden provocar diferentes transiciones de estado de acuerdo con los diagramas de estado de los objetos.
- La transición de estado del objeto muestra entre el estado disponible y el estado destruida puede deberse a dos protocolos diferentes, el protocolo de Registrar Consentimiento (las muestras pueden ser destruidas al registrarse inicialmente el no consentimiento del empleado) o el protocolo de Destruir Muestra (protocolo específico para la destrucción de muestras en el laboratorio).

Por último, en lado inferior izquierdo, también en verde, se encuentra el registro del dato longitudinal de los CRD, con el protocolo existente para realizar la corrección de datos.

Una vez que disponemos del modelo PIM, el siguiente paso es realizar una serie de transformaciones para desarrollar el modelo PSM, con el que se obtiene un modelo específico para la plataforma objetivo de implementación.

En la metodología de la estrategia de diseño, estas transformaciones ya se habían definido y aplicado sobre el patrón PIM, obteniendo el patrón PSM (ver Figura 5-15 del apartado 5.2) de acontecimientos para la implementación de una base de acontecimientos como base de datos relacional. En las siguientes imágenes se muestra el extracto del modelo PSM de ARASIS, acotado al objeto muestra y empleado.

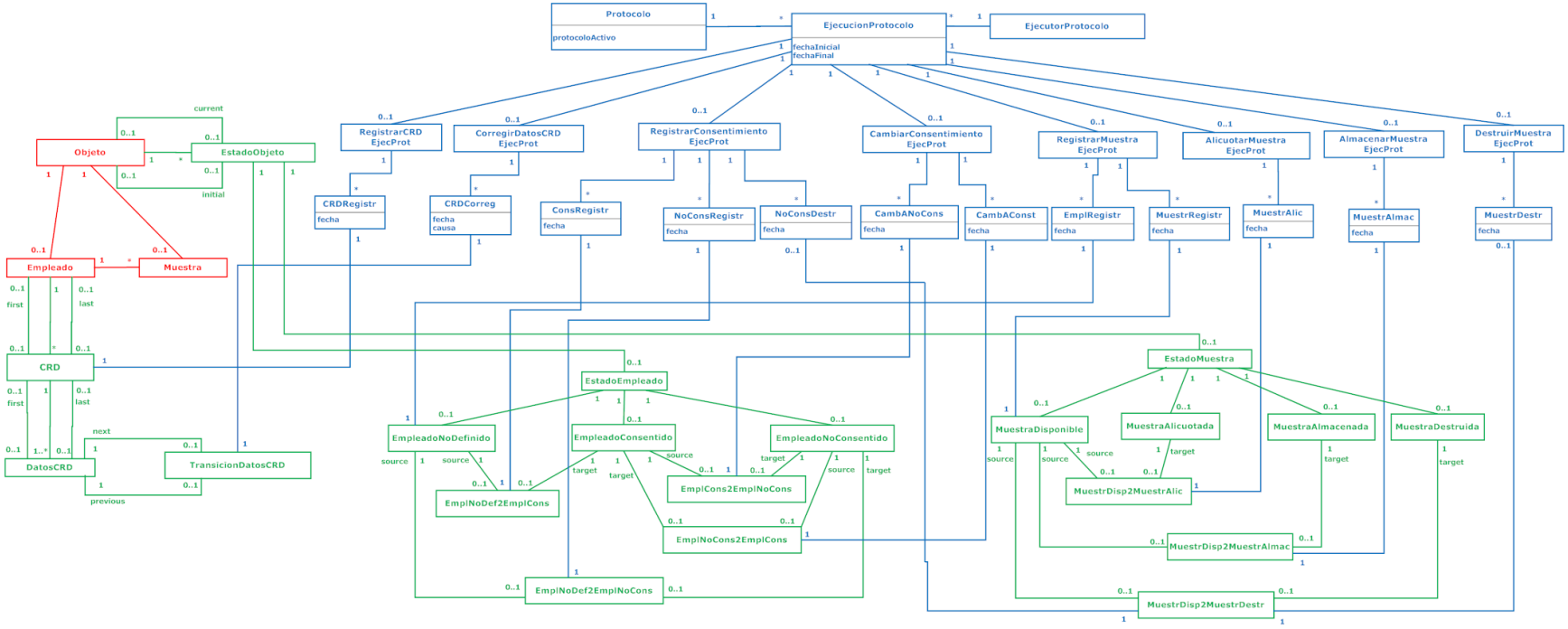


Figura 7-11 Extracto modelo PSM en ARASIS

En la imagen puede verse cómo se respetan todas las relaciones establecidas en el modelo PIM una vez realizadas las transformaciones definidas en la estrategia de diseño.

El modelo PSM es el que se utiliza para construir la OcBase de la arquitectura del SGBAP. En este caso, la plataforma específica para la cual se ha definido el modelo es las bases de datos relacionales, en concreto el SGBD PostreSQL.

Para construir el esquema de la base de datos relacional de la OcBase, aplicando el modelo PSM, se ha utilizado la herramienta DBMain [25], que permite desarrollar modelos conceptuales de bases de datos relacionales, a través de los que obtener el modelo lógico equivalente y construir los posteriores scripts de creación de la base de datos física en el SGBD de PostgreSQL.

En las siguientes imágenes pueden verse tanto el esquema conceptual como el esquema lógico de base de datos construido para ARASIS. En este caso, por motivos de legibilidad de los esquemas, se han acotado a los estados y protocolos del objeto muestra y del dato longitudinal CRD.

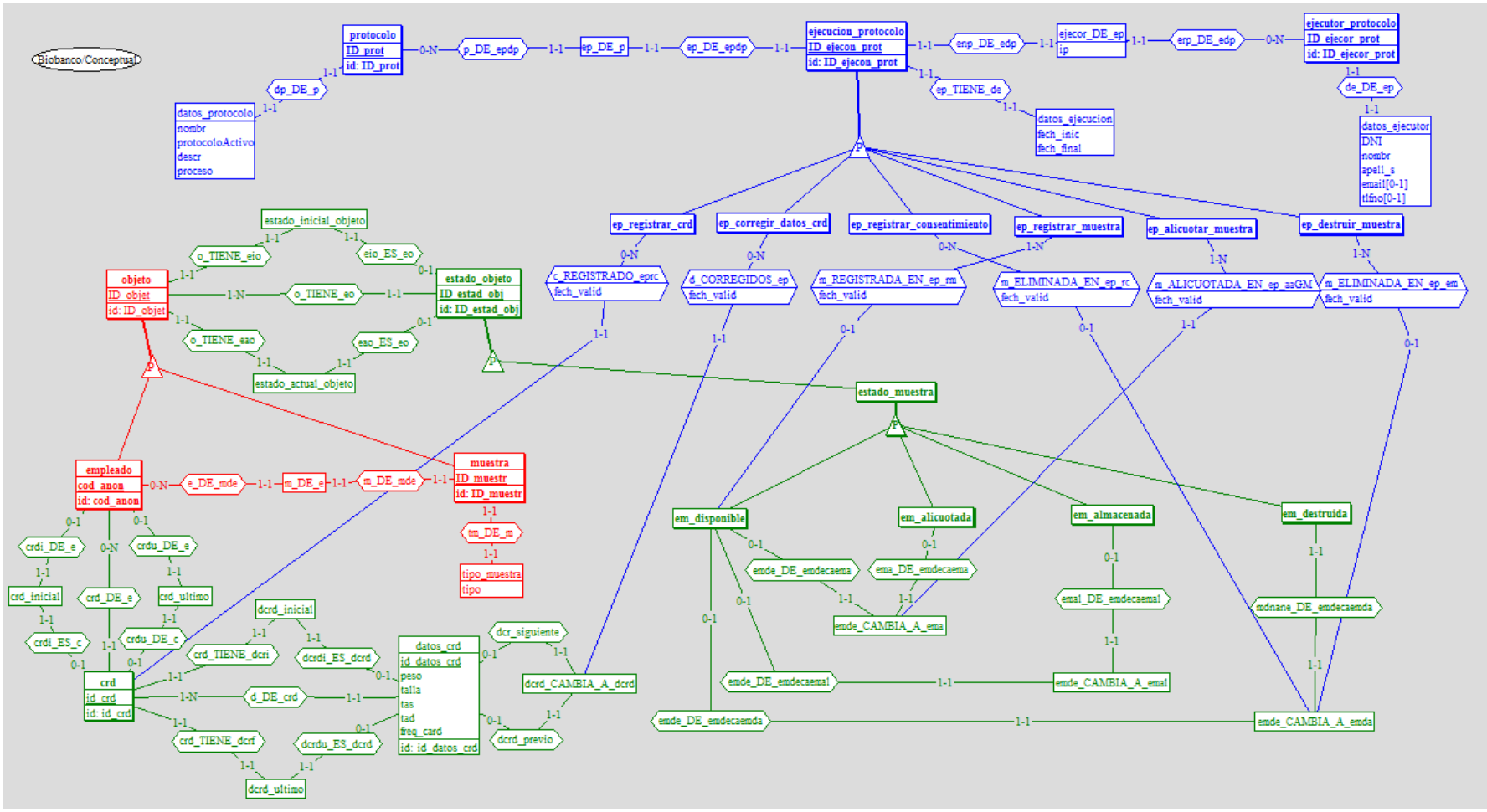


Figura 7-12 Extracto esquema conceptual Base Datos de Base de Acontecimientos de ARASIS

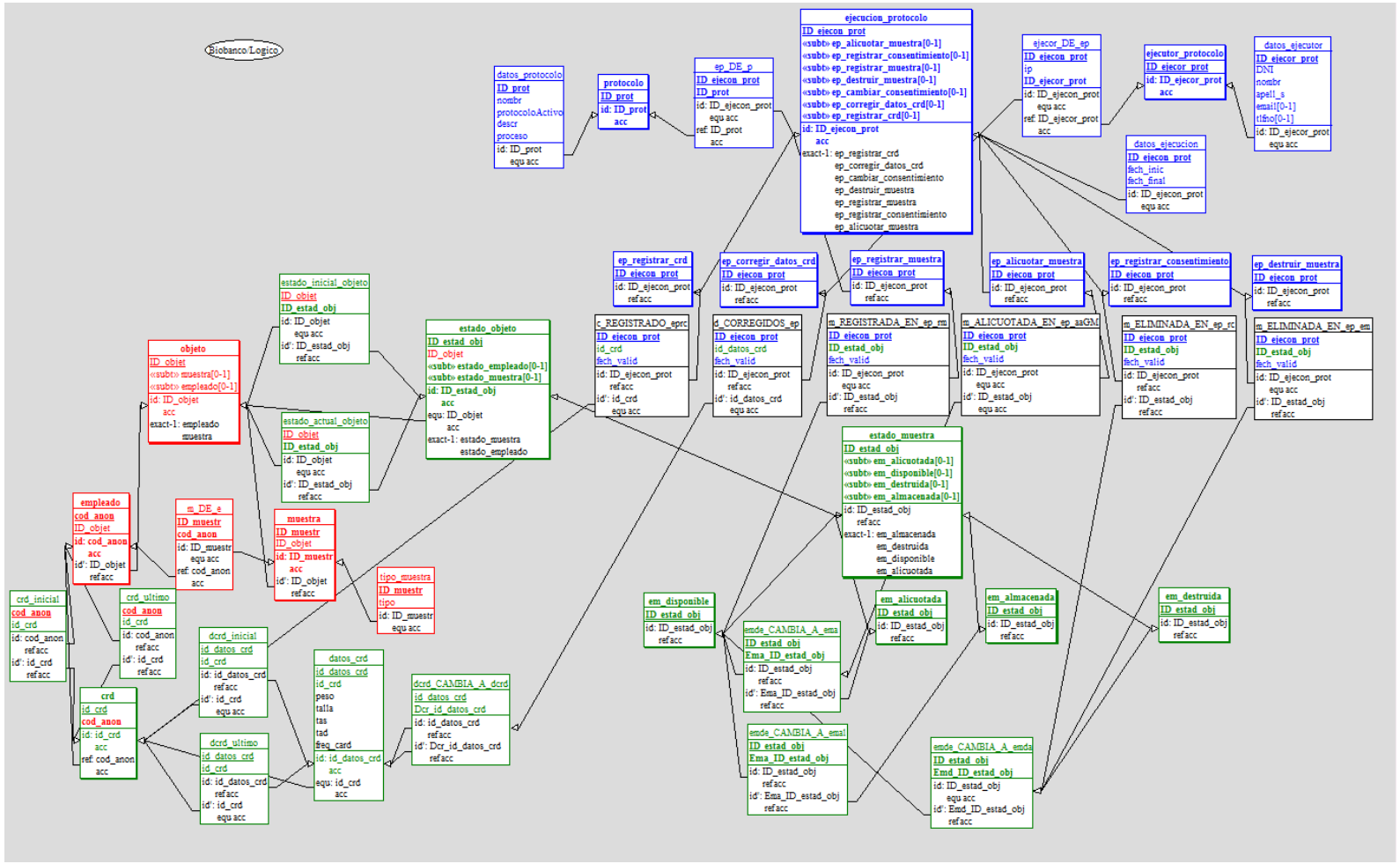


Figura 7-13 Extracto esquema lógico Base Datos de Base de Acontecimientos de ARASIS

La base de datos relacional completa construida contiene más de 900 tablas, siendo la mayor parte las necesarias para incluir todos los objetos, sus estados, sus datos y cada uno de los 66 protocolos, con sus evoluciones según el sistema de versionado, que se ejecutan en el sistema. En el cómputo global de tablas, también se incluyen otras tablas para la utilización de ARASIS como una herramienta más global para la gestión médica de las unidades de salud del proyecto AWHS, proporcionando funcionalidades como impresión de etiquetas o seguimiento de citas de pacientes.

7.1.2 Arquitectura del SGBAP en ARASIS

ARASIS es una herramienta informática construida mediante la arquitectura definida para los SGBAP cuyo objetivo es proporcionar el sistema informático para la gestión del biobanco de muestras biológicas del proyecto AWHS.

A nivel técnico, ha sido desarrollada como aplicación Web, accesible por tanto desde cualquier navegador para los usuarios autorizados a ello, utilizando la tecnología Java para su construcción. En concreto, se ha desarrollado mediante el framework Struts [6], uno de los frameworks de Java para desarrollo Web que implementan el patrón MVC, sobre el que se ha desarrollado la arquitectura del SGBAP, así como otro tipo de tecnologías relacionadas con el desarrollo web -AJAX, CSS, Javascript…- [91][92] y con las utilidades que necesitan los laboratorios en sus protocolos –impresión de etiquetas, lectura de códigos matriciales para los racks…-.

Por necesidades técnicas y requisitos de seguridad del laboratorio establecido en las propias instalaciones de la General Motors, también se ha construido una versión reducida de ARASIS de tipo cliente-servidor (aplicación de escritorio), donde únicamente se incluyen un subconjunto de los protocolos definidos en el proyecto, en concreto, los protocolos de recogida y envío de muestras desde dicho laboratorio.

Sin olvidar, por supuesto, la Base de Acontecimientos basados en Protocolos –OcBase- que, tal y como se ha explicado en el apartado anterior, se ha desarrollado mediante el SGBD PostgreSQL [72].

Como se ha comentado, ARASIS es una herramienta implementada como SGBAP y, por tanto, contiene los elementos establecidos en la arquitectura de un SGBAP, desarrollada en la Figura 5-20 del apartado 5.3.1.

En concreto, se han desarrollado los sistemas:

- Sistema de Protocolos. Este sistema constituye el núcleo de la herramienta, ya que es el encargado de gestionar las ejecuciones de los protocolos y registrar los acontecimientos en la OcBase.
- Sistema OcTrail. Sistema OcTrail de la arquitectura implementado mediante un entorno dinámico a modo de cuadro de mandos que permite visualizar diferentes líneas de vida de los objetos al mismo tiempo.
- Sistema BI. Implementa el sistema Business Intelligence definido en la arquitectura para explotar el conocimiento de la OcBase, mediante la tecnología OLAP, a través de entornos de análisis dinámico de datos y de obtención de indicadores preestablecidos.

Además de los sistemas anteriores, en ARASIS hay implementados otros módulos que

extienden el alcance de un SGBAP para convertirse en una herramienta de gestión global de las necesidades de los equipos médicos del proyecto AWHS, como módulos para el control y seguimiento de citaciones médicas. Aunque la información gestionada en estos sistemas adicionales se enlaza con los objetos de nuestro contexto dentro de la base de datos relacional en la que se implementa la OcBase, no se explica nada de ellos puesto que no entran dentro del alcance de la tesis.

En los siguientes apartados se explican con más detalle los tres sistemas de la arquitectura del SGBAP que se han construido.

7.1.2.1 Sistema de Protocolos

En el Sistema de Protocolos se han implementado los 66 protocolos que componen ahora mismo el conjunto de protocolos que actúan sobre los objetos del sistema en el proyecto AWHS.

Tal como está definido en la arquitectura de un SGBAP, el sistema está construido siguiendo la arquitectura del patrón MVC a través de las componentes Protocol Management Module, Protocol Monitoring Module y la OcBase, donde se guardan todos los acontecimientos generados por la ejecución de los protocolos.

En la siguiente figura puede verse con más detalle la arquitectura implementada para el Sistema de Protocolos:

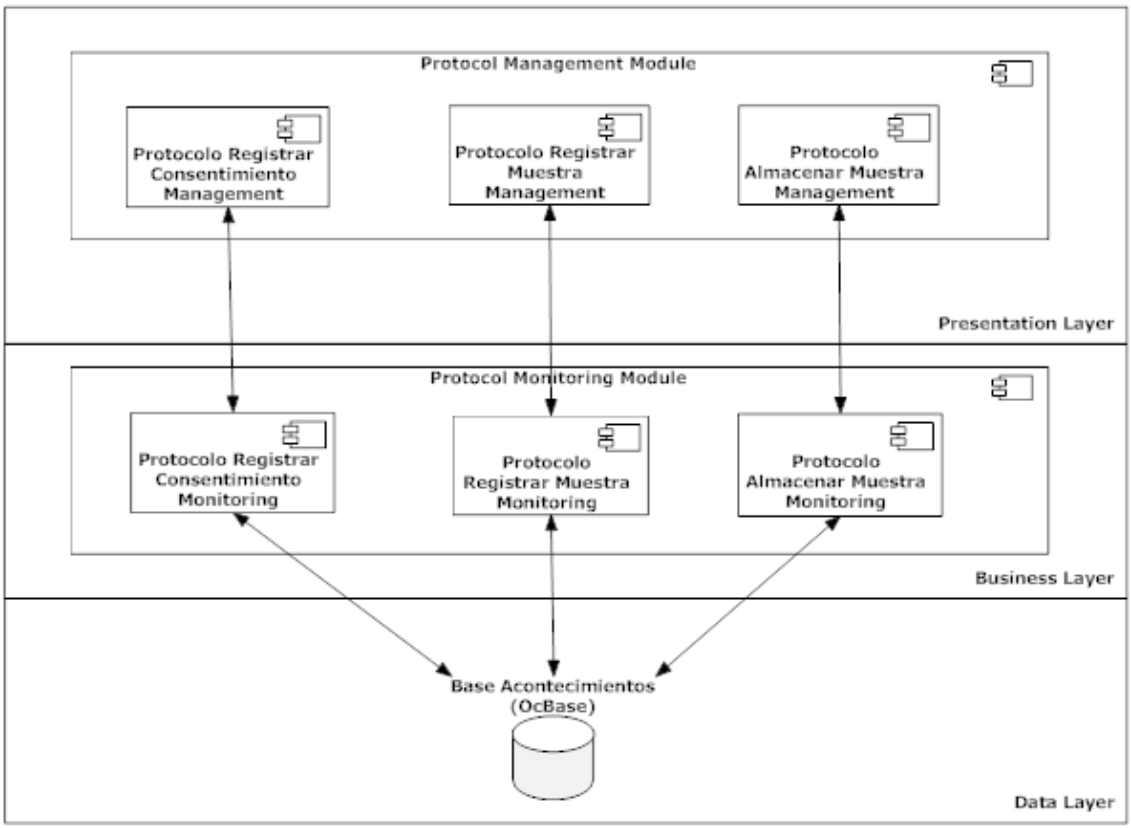


Figura 7-14 Arquitectura Sistema Protocolos ARASIS

Como se puede ver en la imagen, para cada protocolo se desarrolla una subcomponente dentro de cada capa definida en la arquitectura SGBAP. En la imagen se han incluido a modo de ejemplo 3 protocolos de los 66 existentes en ARASIS.

En la siguiente imagen se desciende un nivel más en el detalle de las subcomponentes que se desarrollan para cada protocolo, tomando como ejemplo el protocolo 'Registrar muestra':

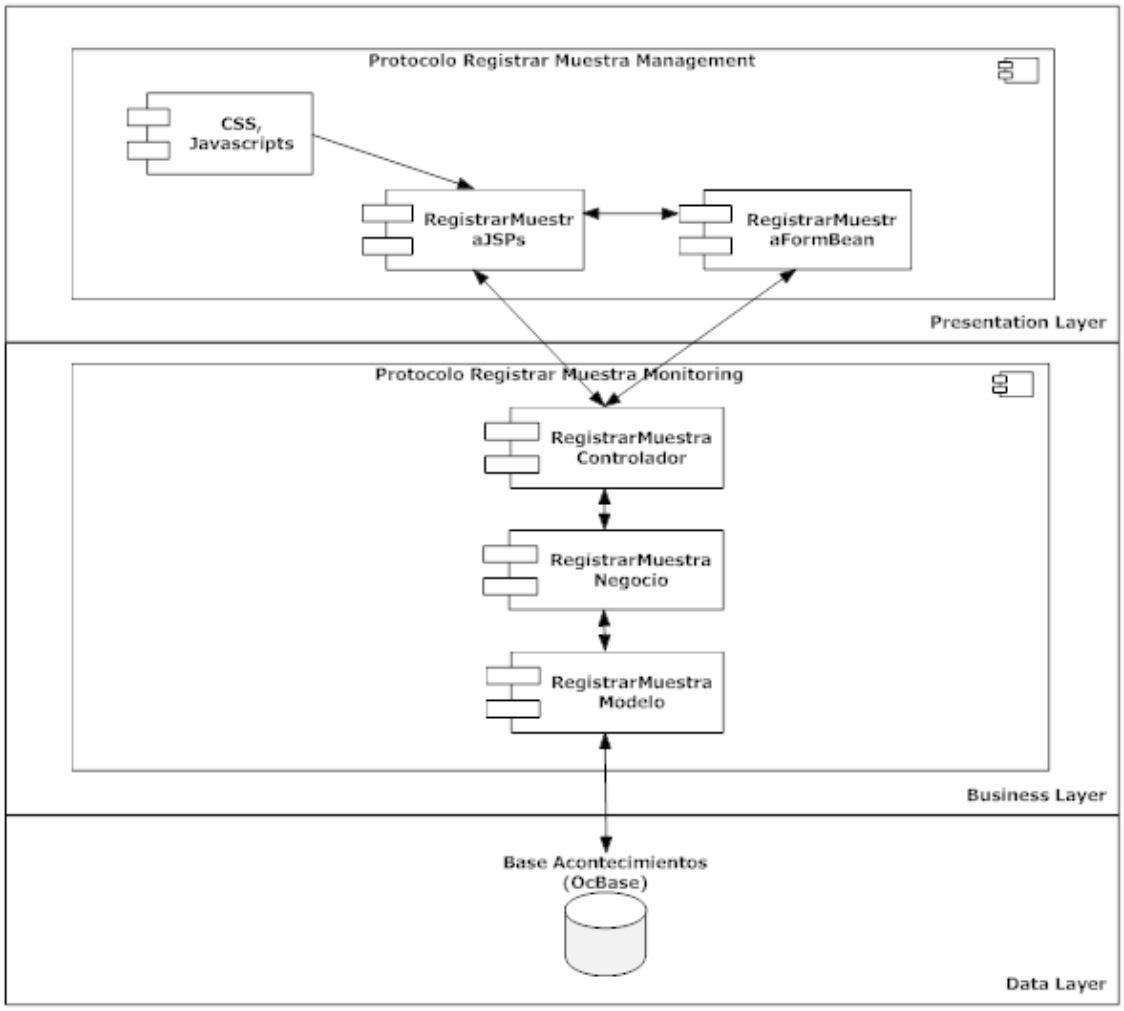


Figura 7-15 Arquitectura en detalle Protocolo ARASIS

De esta forma, para cada protocolo se desarrollan las siguientes subcomponentes:

- Capa de presentación, **Protocol Management Module**. Incluye los siguientes elementos:
  - Las páginas web en lenguaje de programación JSP que proporcionan la interfaz de usuario, junto con los ficheros CSS y Javascript, los cuales proporcionan los estilos de visualización y funciones dinámicas de gestión de la interfaz respectivamente.
  - Las clases Java que recogen y gestionan los datos de los formularios que se rellenan como evidencia de la ejecución de los protocolos, llamados 'FormBean' en el framework Struts, y que constituyen los elementos de comunicación con la capa

negocio del patrón MVC de la arquitectura.

- Capa negocio, **Protocol Monitoring Module**. Contiene la lógica para la ejecución de todos los protocolos implementados en ARASIS. Para cada protocolo se dispone de:
  - Controlador de gestión, **Protocolo_k Controlador**, siendo Protocolo_k. el protocolo en cuestión ('RegistrarMuestra' en el ejemplo). Clase Java del framework Struts, llamado ActionForm, que enlaza con la capa Vista recibiendo y enviando los elementos FormBean para la comunicación con los JSP de la interfaz.

    La clase controlador utiliza la clase de negocio para realizar toda la lógica que necesita en su funcionamiento.
  - Clase Java de lógica de negocio, **Protocolo_k Negocio**. En esta clase se gestiona toda la lógica de negocio del protocolo, incluyendo la gestión de los acontecimientos con la OcBase a través de sentencias SQL (preparadas en la clase modelo) mediante conexiones JDBC –interfaz de conexión en Java con sistemas gestores de bases de datos como PostgreSQL-.
  - Clase Java de modelo, **Protocolo_k Modelo**. En esta clase se definen las sentencias SQL que se utilizan en la clase de negocio para la consulta y registro de la información de la ejecución de los protocolos –acontecimientos basados en protocolos-, implementando las conexiones JDBC a la OcBase.
- Capa de datos, **OcBase**. En el Sistema de Protocolos se realiza el registro de todos los acontecimientos que generan las ejecuciones de los protocolos en la Base de Acontecimientos basados en Protocolos del sistema ARASIS, la OcBase.

El objetivo de esta arquitectura tan modular del Sistema de Protocolos es aprovechar al máximo la escalabilidad que proporciona el patrón MVC en el desarrollo de aplicaciones. De esta forma, en caso de que haya que corregir o modificar un protocolo únicamente hay que modificar los elementos anteriores construidos para el protocolo en cuestión, sin modificar absolutamente nada del resto de protocolos implementados en el sistema, minimizando al máximo la posibilidad de generación de errores involuntarios en el resto de protocolos.

Por otro lado, añadir nuevos protocolos es sumamente sencillo, ya que únicamente implica definir los elementos de la arquitectura para un protocolo y añadirlos al sistema de forma transparente al resto de protocolos, lo que maximiza la escalabilidad del sistema. Además, esta independencia también permite que cada protocolo se ajuste a sus necesidades particulares, permitiendo diferentes interfaces según sea cada situación. En la siguiente imagen se muestra un ejemplo de la interfaz del sistema de protocolos de ARASIS:

Figura 7-16 Ejemplo de Interfaz Sistema de Protocolos ARASIS

Por último, como se ha comentado en el apartado 3.3 en el que se describía de forma abstracta un SGBAP, uno de los mecanismos de explotación del conocimiento almacenado en forma de acontecimientos en el sistema debe ser la capacidad de recuperar la información de cualquier acontecimiento registrado en el sistema. En el caso de ARASIS, esta funcionalidad se ha implementado mediante la funcionalidad llamada 'Búsqueda transversal', que permite establecer unas condiciones de búsqueda para consultar todos los acontecimientos que cumplen con esas condiciones para cada uno de los protocolos del sistema. En la siguiente imágen podemos ver un ejemplo:

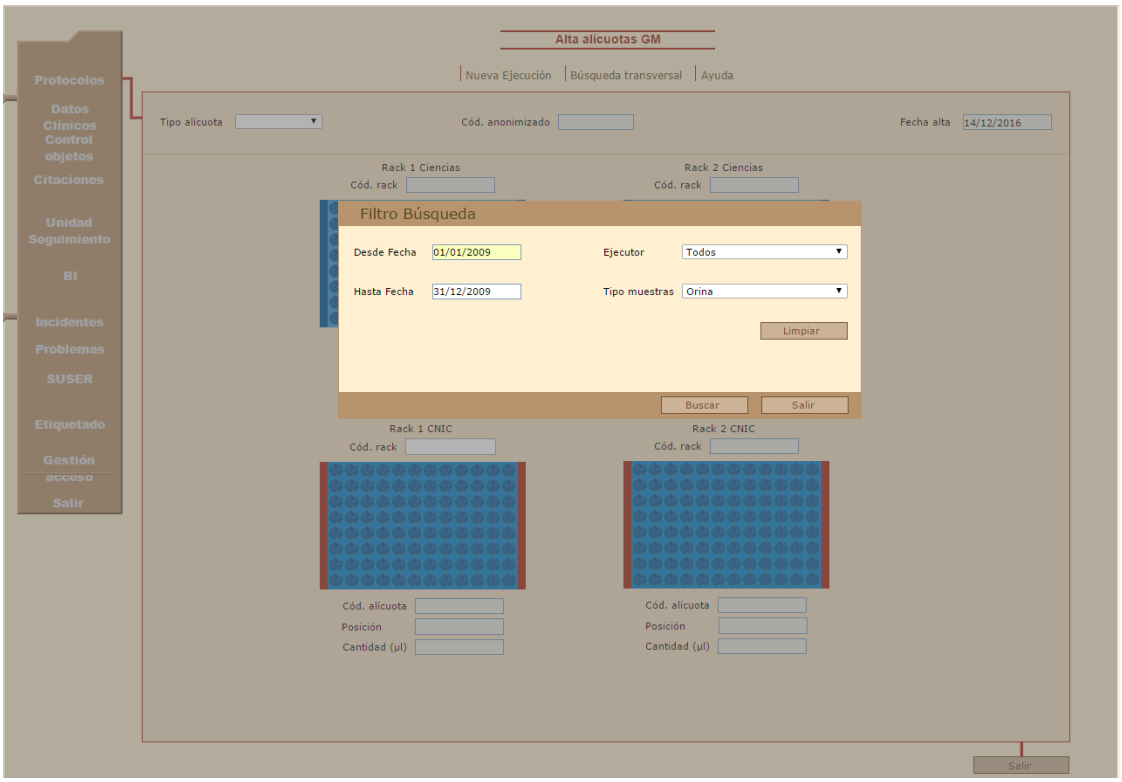


Figura 7-17 Inicio búsqueda transversal en el Sistema de Protocolos

En este caso, podemos establecer un rango de fechas de ejecución de los protocolos que queremos buscar, un ejecutor de protocolo concreto que ha realizado dichas ejecuciones y condiciones particulares según el protocolo en el que se realiza la búsqueda. En el ejemplo, buscamos las ejecuciones del protocolo 'Alta alícuotas GM' que se realizaron en el año 2009, para cualquier ejecutor posible en el sistema ARASIS y para los tipos de muestra de 'Orina', como condición particular de este tipo de protocolo. Como resultado de esta búsqueda de ejemplo se obtienen 3400 ejecuciones del protocolo, o lo que es lo mismo, 3400 acontecimientos en los que se ejecuta el protocolo 'Alta Alícuota GM', pudiendo consultar la información de cada acontecimiento resultado de la búsqueda.

7.1.2.2 Sistema OcTrail

El Sistema OcTrail de la arquitectura SGBAP en ARASIS, identificado como 'Working Business Environment' en la interfaz de usuario, proporciona un entorno dinámico e interactivo en el que poder consultar la red de líneas de vida de los objetos del sistema descrita en el apartado 3.3 de descripción del SGBAP, permitiendo consultar la historia de acontecimientos de un objeto.

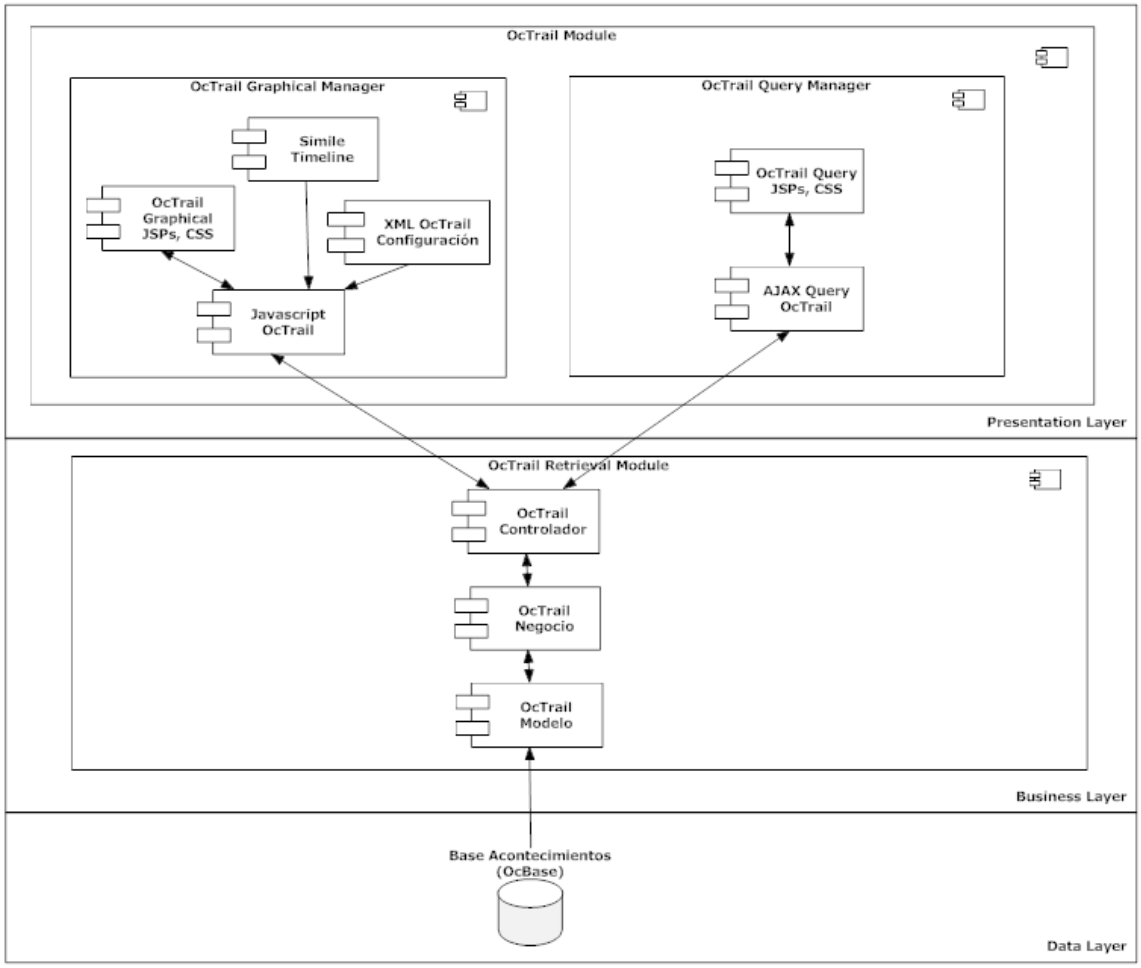


Figura 7-18 Arquitectura detallada del Sistema OcTrail en ARASIS

La arquitectura detallada del sistema OcTrail implementado en ARASIS (imagen anterior), está constituida por las siguientes componentes:

- Componentes de la capa de presentación, **OcTrail Module**. Siguiendo la arquitectura SGBAP, el OcTrail Module se compone del OcTrail Graphical Manager –la componente de visualización del entorno- y el OcTrail Query Manager –componente de búsqueda para filtrar datos-:
  - **OcTrail Graphical Manager**. Esta componente está formada por las páginas web en lenguaje de programación JSP que proporcionan la interfaz de usuario, junto con los ficheros CSS y Javascript que permiten construir el entorno dinámico.

    Dentro de los elementos Javascript se encuentra la librería Simile Timeline [79], la cual proporciona un widget Javascript –un pequeño aplicativo gráfico en tecnología Javascript integrable en aplicaciones web- que permite representar gráficamente líneas de tiempo con los eventos que han sucedido en un periodo determinado. Veremos más en detalle esta librería y cómo se utiliza en el apartado Apéndice B.

    En ARASIS estas líneas de tiempo son llamadas 'líneas de vida', ya que muestran precisamente eso, la línea de vida de un objeto siendo los acontecimientos los eventos que se representan.

El entorno utiliza la tecnología AJAX, en el elemento Javascript OcTrail, para comunicarse con los controladores de la capa de negocio del patrón, realizando consultas para la obtención de la información de los acontecimientos que van a ser representados en las líneas de vida.

El fichero de configuración XML almacena la situación actual de un usuario en el entorno –líneas de vida visualizadas- para ser cargado en las mismas condiciones en siguientes inicios de sesión en el entorno.

- **OcTrail Query Manager**. Conjunto de archivos JSP, CSS y Javascript que permiten recoger las condiciones establecidas en los filtros de búsqueda del entorno dinámico y lanzar las consultas, mediante tecnología AJAX, a los controladores de la capa negocio del patrón para obtener las líneas de vida resultado de los filtros.

  En la siguiente imagen se puede ver los dos componentes en la interfaz.

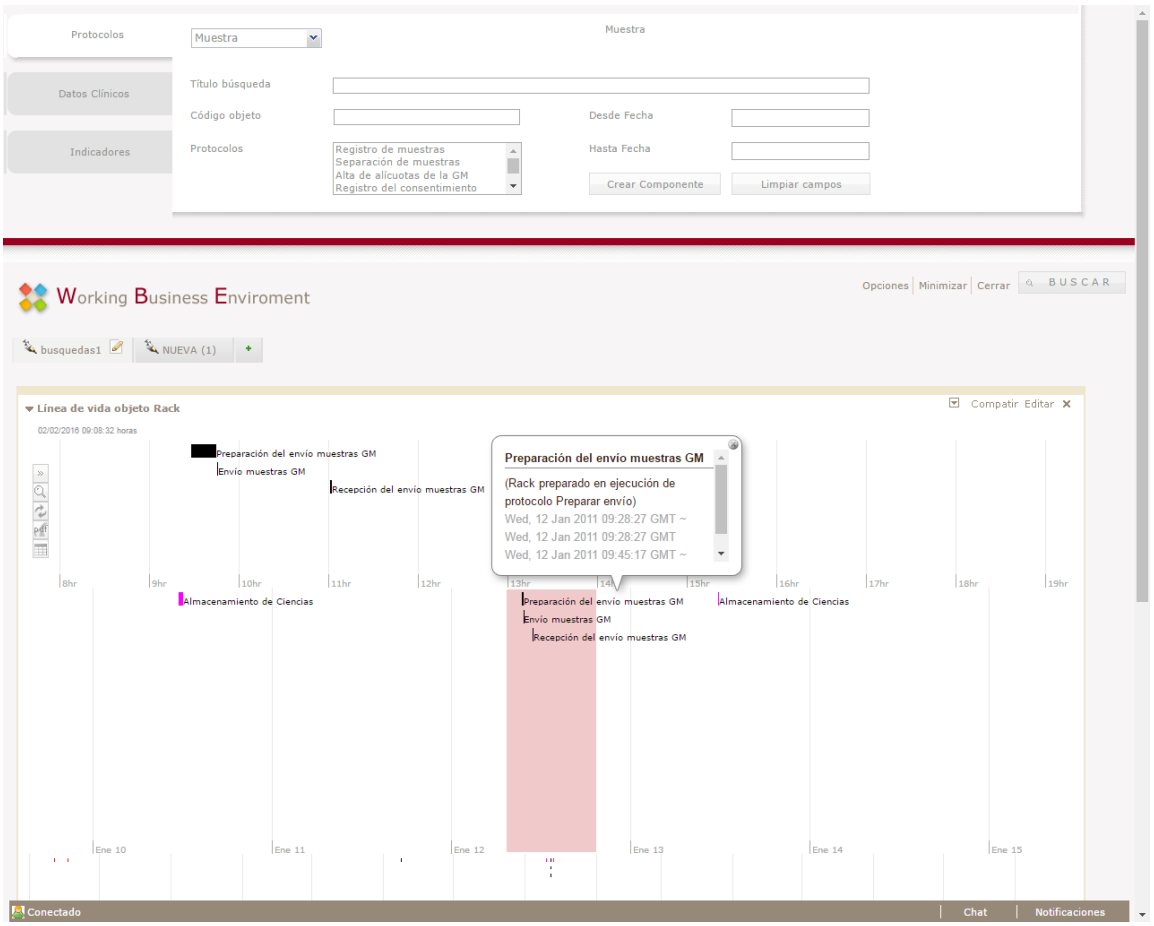


Figura 7-19 Interfaz del OcTrail Module en ARASIS

En la parte superior se encuentra la interfaz del OcTrail Query Manager, donde se introducen las condiciones de búsqueda, y en la parte inferior la interfaz del entorno dinámico proporcionado por el OcTrail Graphical Manager.

- Componentes de la capa negocio, el **OcTrail Retrieval Module**. En esta capa se construyen los siguientes elementos:

- Controlador de gestión **OcTrail Controlador**. Clase Java del framework Struts, en este caso mediante elementos Servlets, que enlaza con la capa de presentación recibiendo las peticiones AJAX del OcTrail Query Manager y del OcTrail Graphical Manager.

  Las peticiones recibidas las envía a la lógica de negocio –OcTrail Negocio- para que sean procesadas, recogiendo los resultados y enviando las respuestas a la capa de presentación.

  En estas comunicaciones AJAX, se utiliza el lenguaje XML para el formato de las entradas y salidas.
- Clase Java de negocio, **OcTrail Negocio**. En esta clase se gestiona toda la lógica de negocio del sistema OcTrail, recibiendo las peticiones desde el controlador del sistema y transformándolas, cuando es necesario, en solicitudes de información a la OcBase, a través de la clase modelo, para obtener los acontecimientos y demás información que se requiere representar en el entorno gráfico del sistema.

  Entre otros cometidos, en la lógica de negocio se preparan los acontecimientos en formato JSON [92]–formato Javascript para el intercambio de datos-, para poder ser representados por la librería Simile Timeline en la representación de las líneas de vida. Como hemos comentado anteriormente, en el apartado Apéndice B se explica más en detalle el funcionamiento de la librería.
- Clase Java de modelo, **OcTrail Modelo**. En esta clase se definen las sentencias SQL que se utilizan en la clase de negocio para la obtención de las líneas de vida de los objetos.

  Las sentencias SQL se construyen de forma dinámica mediante un motor de construcción de consultas SQL para permitir incluir cualquier combinación posible en las selecciones realizadas en los filtros de búsqueda.

- Capa de datos, **OcBase**. El Sistema OcTrail utiliza como fuente de información los acontecimientos registrados en la OcBase por el Sistema de Protocolos.

En los apéndices se explica en detalle el entorno gráfico del OcTrail Graphical Manager, presentando: (1) el funcionamiento dinámico del conjunto del entorno (ver apartado A.1 del apéndice A), (2) la explicación de cómo el sistema OcTrail proporciona el acceso a la red de líneas de vida de los objetos –la historia- (ver apartado A.2 del apéndice A) y (3) el caso particular de las líneas de vida de datos longitudinales (ver apartado A.3 del apéndice A).

7.1.2.3 Sistema BI, sistema de Business Intelligence

En el Sistema BI de la arquitectura del SGBAP en ARASIS, como veremos a continuación, se han construido varios entornos para explotar el conocimiento almacenado en la OcBase de formas diferentes, utilizando la tecnología OLAP especificada en la arquitectura del SGBAP para ello.

Estos entornos se han construido según el patrón MVC establecido en la arquitectura del SGBAP, como se puede ver en la siguiente imagen de la arquitectura del sistema BI.

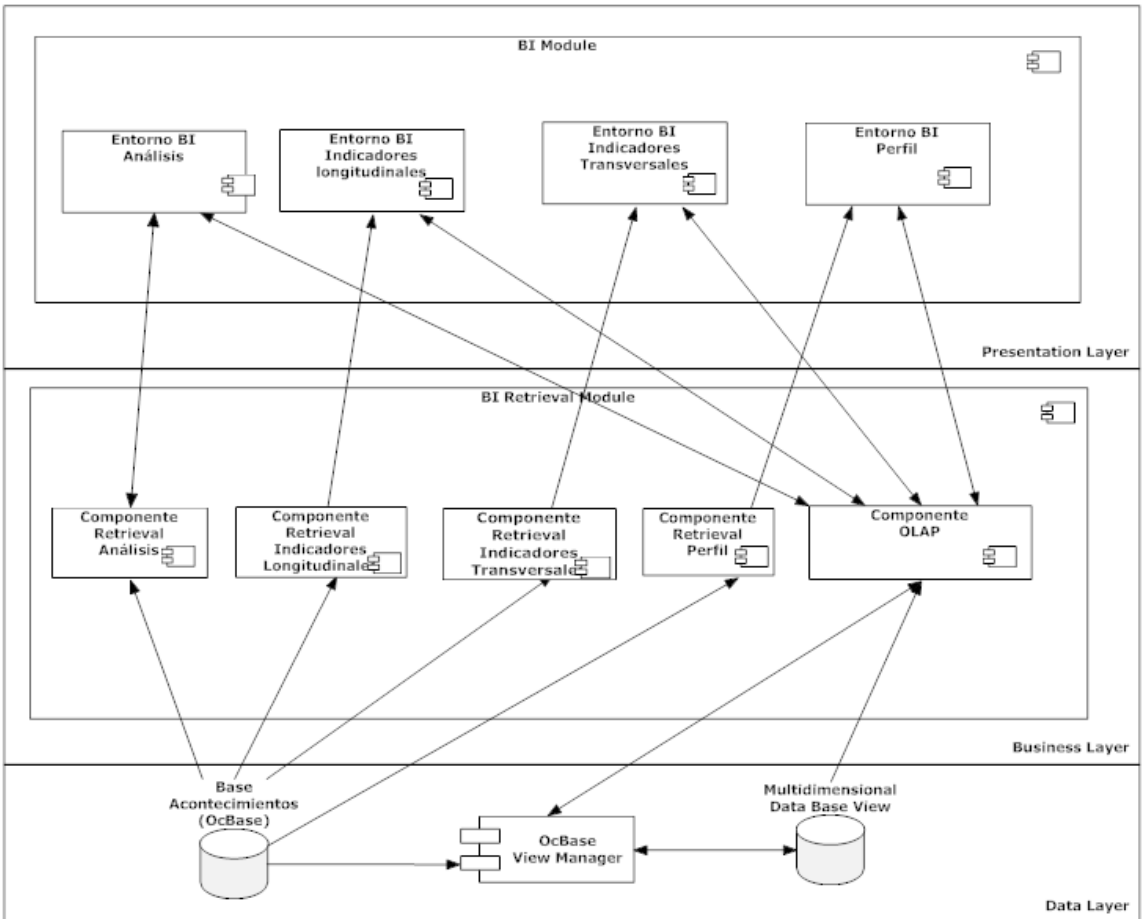


Figura 7-20 Arquitectura general del Sistema BI en ARASIS

Existen cuatro entornos diferentes:

- Análisis de datos, **Entorno BI Análisis**. Permite un análisis dinámico de los cubos OLAP definidos en el sistema BI. Este entorno nos permitirá conocer, por ejemplo, qué protocolo es el que más se ejecuta en el sistema y dinámicamente cambiar la consulta para restringir el tiempo en el que queremos conocer el dato, desglosando el resultado por años y profundizar en un año concreto para conocer las ejecuciones de protocolos por meses del año.
- Indicadores longitudinales, **Entorno BI Indicadores Longitudinales**. Permite obtener la evolución en el tiempo de ciertos indicadores predefinidos que se obtienen mediante los cubos OLAP. Por ejemplo, conocer la evolución del número de muestras registradas durante el tiempo, o la evolución del peso medio de los empleados registrados en sus datos clínicos.
- Indicadores transversales, **Entorno BI Indicadores Transversales**. Permite obtener indicadores que miden un dato en un momento puntual del tiempo a través de la información proporcionada por los cubos OLAP. Por ejemplo, conocer el número de ejecuciones de un protocolo o la distribución de sexos de los empleados de la cohorte en una fecha determinada.
- Perfil del biobanco, **Entorno BI Perfil**. Permite utilizar la información de los cubos

OLAP para obtener el perfil de la cohorte del biobanco. El perfil de la cohorte es una descripción de un conjunto de características que describe a los empleados que la conforman, como por ejemplo, el empleado característico de la cohorte es un varón, no fumador, entre 40 y 55 años y con un peso medio de 72 kilogramos.

El desglose de los entornos se produce en la capa de presentación del patrón MVC, con la correspondiente lógica para cada interfaz, utilizando en todos los entornos el mismo sistema OLAP, y sobre todo, el mismo sistema multidimensional construido en la capa de modelo para nutrir de información a los cubos OLAP. De esta forma se rentabiliza al máximo el sistema OLAP desarrollado.

En los cubos OLAP se utiliza la información de todos los acontecimientos basados en protocolos registrados en la OcBase, tanto de los protocolos cuyo efecto son transiciones de estado como de aquellos cuyo efecto son el registro de datos longitudinales. Por tanto, al igual que en el sistema OcTrail, en el sistema BI se realiza un explotación integral de la información registrada en los acontecimientos.

En la siguiente imagen se muestra en un nivel de detalle mayor la arquitectura del sistema BI centralizando el foco en el entorno de análisis de datos:

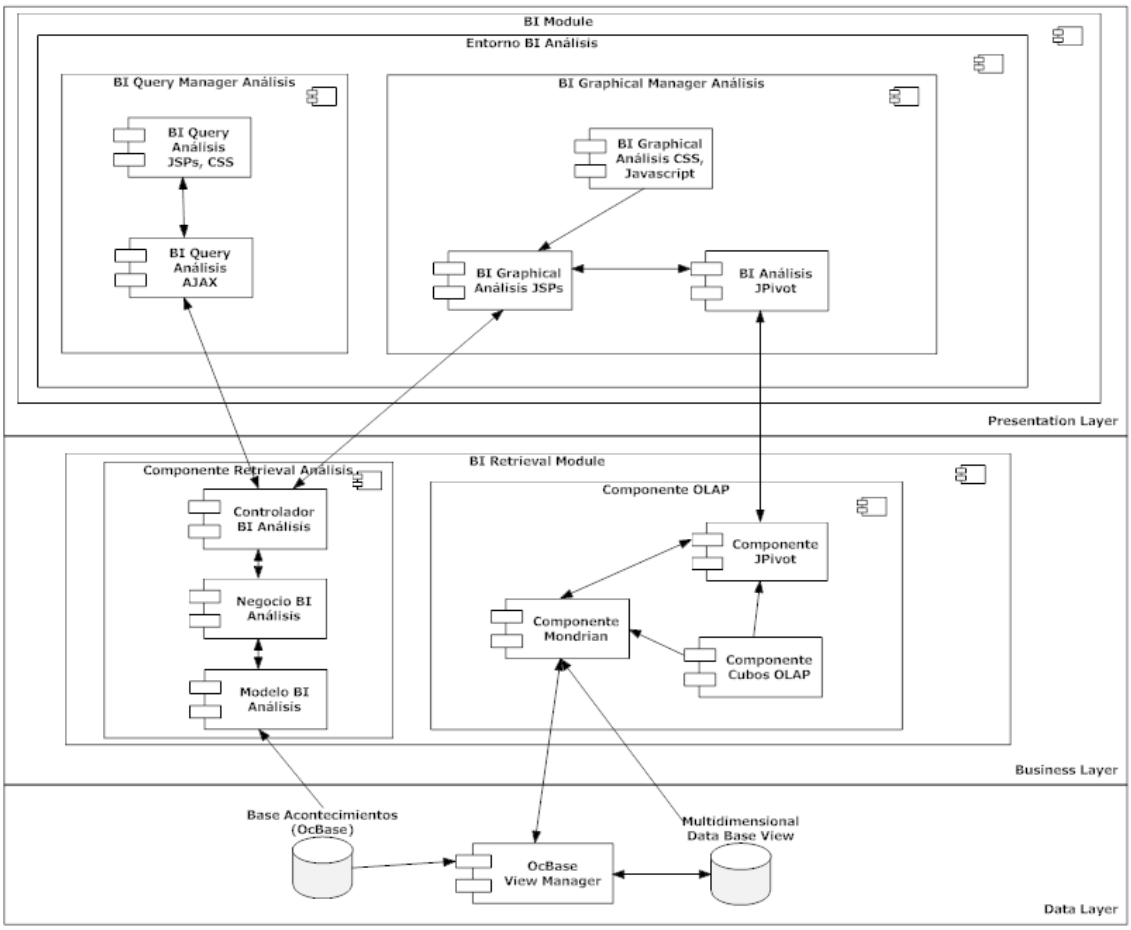


Figura 7-21 Arquitectura detallada del Sistema BI en ARASIS – Entorno análisis de datos

Como se puede ver en el detalle de la arquitectura, se desarrollan las siguientes subcomponentes:

- Para cada entorno, en la capa de presentación, se construye un 'BI Module' con los siguientes elementos:
  - En el **BI Query Manager**, se encuentran el conjunto de archivos JSP, CSS y Javascript que permiten seleccionar las condiciones iniciales con las que se comienza a explotar los cubos OLAP a través del BI Graphical Manager.
  - **BI Graphical Manager**. Conjunto de archivos JSP, CSS y Javascript que permiten utilizar la librería JSP JPivot [46], librería que nos proporciona el uso dinámico de los cubos OLAP realizando las peticiones a la componente JPivot interna del BI Retrieval Module.

    La inicialización de esta componente se realiza mediante una comunicación del BI Controller del entorno utilizando las selecciones realizadas en el BI Query Manager por el usuario.

  Las particularidades de cada uno de los cuatro entornos en cuanto a interfaz gráfica se explican en detalle en el VI.
- Capa negocio, **BI Retrieval Module**. En esta capa se construyen los siguientes elementos:
  - **Componente Retrieval** del entorno. Cada entorno tiene definido en la capa de negocio los elementos habituales de esta capa en la arquitectura SGBAP:
    - **Controlador BI**, controlador particular de cada entorno de gestión. Clase Java del framework Struts, construido como Servlet, que enlaza con la capa de presentación recibiendo las peticiones AJAX del BI Query Manager para inicializar cada entorno BI según las selecciones realizadas por el usuario.

      Para ello, envía a la componente BI Graphical Manager la consulta MDX inicial que equivale a las selecciones de usuario para que sea utilizada por la librería JPivot y comenzar así la explotación de los cubos OLAP.

      Una vez iniciado el entorno, todas las peticiones que realiza el usuario en el BI Graphical Manager van dirigidas directamente a las componentes JPivot y Mondrian del BI Retrieval Module.
    - **Negocio BI**, lógica de negocio del entorno. Clase Java que contiene toda la lógica de negocio de cada entorno BI del sistema, incluyendo la preparación de las consultas MDX iniciales para el inicio del sistema OLAP y la petición al modelo de cualquier tipo de información que se requiera en la interfaz (solicitada por el controlador).
    - **Modelo BI**, enlace con la OcBase de cada entorno. Clase Java que implementa las conexiones JDBC con la OcBase para la obtención de información necesaria en la interfaz, como los nombres de los protocolos existentes para la preparación de las selecciones del BI Query Manager.
  - **Componente OLAP**. Esta componente contiene toda la lógica necesaria para el funcionamiento de la tecnología OLAP en todos los entornos BI del sistema ARASIS, y está compuesto de las siguientes subcomponentes:
    - **Componente JPivot**. Implementa la lógica de negocio de la librería JPivot. Las acciones que realiza el usuario en la interfaz las transforma en consultas MDX que

envía a la componente Mondrian para lanzarlas contra los cubos OLAP.

Posteriormente, recibe los resultados de la componente Mondrian y los envía directamente a la componente BI Graphical Manager para ser representados en la interfaz del usuario.

- **Componente Mondrian**. Implementa la lógica de negocio de la librería Mondrian [70], la cual transforma las consultas MDX en consultas SQL extrayendo los datos del Multidimensional Data Base View mediante la definición realizada en los cubos OLAP construidos en la componente Cubos OLAP.
- **Componente Cubos OLAP**. Contiene todos los cubos OLAP construidos para el funcionamiento del sistema BI en ARASIS. Los cubos OLAP son construidos en formato XML, y los utilizan tanto la componente JPivot (para conocer la estructura del cubo, poder representarla en la interfaz, y construir las consultas MDX) como la componente Mondrian (para traducir las consultas MDX en consultas SQL a la OcBase).

  Como referencia de la cobertura del sistema BI, se han construido un total de 24 cubos OLAP que permiten analizar cada objeto del sistema y realizar las consultas de los indicadores de los diferentes entornos.

En el Apéndice C se dispone de una explicación en detalle de la utilización de estas librerías en el sistema BI.

- Capa de datos, componente **OcBase View Manager** y la base de datos multidimensional resultante **Multidimensional Data Base View**. La tecnología OLAP se sustenta bajo la filosofía de bases de datos multidimensionales, tal y como se ha explicado en el desarrollo de la arquitectura del SGBAP en el apartado 5.3.4. De esta forma, para el funcionamiento de los cubos OLAP se requiere disponer de los datos que conforman la tabla de hechos y las dimensiones en un esquema multidimensional de datos en forma de estrella o copo de nieve.

  Para adaptar la estructura de la OcBase de ARASIS a estos requerimientos, se ha desarrollado la componente 'OcBase View Manager' con la que se construyen dinámicamente las vistas necesarias para el funcionamiento de los cubos OLAP, las cuales se representan mediante la componente 'Multidimensional Data Base View'. En total, para el funcionamiento de los 24 cubos OLAP desarrollados, se requiere de la construcción de 79 vistas SQL diferentes.

En el apéndice A se explican con más detalle el funcionamiento de los cuatro entornos, como muestra de la posibilidad de generar diferentes vistas con los mismos datos que nos permite la arquitectura del SGBAP, y se describe como se definen y construyen un par de cubos OLAP a modo de ejemplo (apartados del A.4 al A.8 del VI).

7.2 Análisis y evaluación de la implementación de ARASIS

La evaluación de la implementación de ARASIS en base a la estrategia de diseño, la realizaremos desde dos puntos de vista:

- La valoración de la propia implementación de la estrategia, analizando las ventajas y desventajas de su utilización para construir sistemas de monitorización de procesos.

- La evaluación del rendimiento de la herramienta en términos de tiempo y espacio de almacenamiento.

Ventajas y desventajas de utilizar la estrategia de diseño

En primer lugar, una de las principales ventajas de la utilización de la estrategia es que el diseño del sistema a construir se realiza mediante la aplicación de patrones de diseño, los patrones de acontecimientos, que se repiten a modo de bloques de construcción para los diferentes protocolos que existan en los procesos de negocio. El uso de patrones de diseño, como es bien conocido en el desarrollo de software, proporciona múltiples beneficios como [54]: la obtención de elementos reutilizables en el diseño, mayor facilidad de aprendizaje y comunicación entre los diseñadores al disponer de un vocabulario y conocimiento establecido y común, o la estandarización del diseño dentro del entorno de aplicación de los patrones [34].

En este caso, además de los beneficios anteriores, la arquitectura planteada proporciona una gran flexibilidad como hemos visto para la evolución de protocolos, siendo relativamente sencilla la inclusión o modificación de los protocolos existentes debido a:

- La utilización de los patrones. Un nuevo protocolo sería la aplicación del patrón de acontecimientos correspondiente.
- La modularidad en la arquitectura. Un nuevo protocolo implica añadir sus componentes específicas dentro de la arquitectura sin afectar al resto de protocolos existentes.

Como consecuencia de esto, las modificaciones pueden realizarse más rápidamente y con menor margen de error.

La construcción de ARASIS se ha realizado de forma totalmente escalable a medida que se definían nuevos protocolos en el laboratorio, o se ampliaba el alcance del sistema para gestionar más procesos del biobanco y de los departamentos de salud involucrados en el proyecto AWHS. Todo ello con el evidente cambio en los equipos de desarrolladores y diseñadores a lo largo del tiempo, sin ninguna merma en las características y calidad del sistema.

Además, la utilización del patrón MVC para la estructura de la arquitectura posibilita la construcción de los diferentes módulos como sistemas distribuidos. En ARASIS, se implementó de forma distribuida una parte del sistema dentro de las instalaciones de la GM para su uso exclusivo dentro del laboratorio.

Sin embargo, precisamente esta alta modularidad que proporciona las ventajas anteriores, también nos provoca la mayor desventaja que tiene la estrategia diseñada. En el caso de la base de acontecimientos, el hecho de que cada objeto, protocolo y estado, con sus correspondientes transiciones y relaciones, tengan su propia clase UML en los modelos PIM y PSM que se construyen, traducidas en tablas en la base de datos relacional final, provoca que el tamaño final de estos modelos llegue a tener un tamaño considerable que puede dificultar su gestión. En el caso de ARASIS, como se comentó en la explicación de la implementación, la base de datos relacional final tiene más de 900 tablas físicas, de las cuales buena parte son consecuencia de los 66 protocolos implementados, todos los posibles estados de los objetos y sus relaciones.

Evidentemente, en los modelos PIM y PSM estamos hablando de tamaños similares en cuanto

a clases UML definidas en ellos. Aunque las herramientas de modelado y diseño no tienen problemas para soportar estos tamaños de trabajo, se hizo necesaria la división de los modelos en varios submodelos para facilitar la gestión de los mismos por los diseñadores.

En el caso del propio software construido pasa algo similar. La implementación de forma modular de los protocolos implica la existencia de múltiples ficheros de código para desarrollar la lógica de todos los protocolos. Sin embargo, la complejidad de cada fichero de código se reduce considerablemente en comparación a si se hubiera construido de una forma más integrada y compacta, reduciéndose además cada protocolo a un conjunto muy limitado de ficheros, por lo que la complejidad final para el mantenimiento es menor en su conjunto.

Por otro lado, en el caso particular de los patrones PIM y PSM, permiten al diseñador mantener la consistencia entre los modelos, además de obtener modelos que simplifican el trabajo necesario en caso de cambiar la plataforma de implementación. No obstante, en el caso de situaciones como ARASIS donde los modelos son considerablemente grandes, la transformación del PIM al PSM es costosa debido a que las transformaciones hay que realizarlas manualmente. El desarrollo de un proceso automático para la transformación del modelo PIM al PSM es una tarea para el trabajo futuro.

Por último, la construcción del sistema ARASIS como SGBAP ha posibilitado cumplir los requisitos en cuanto a disponibilidad y persistencia de la información de las muestras biológicas que se le exigen a un biobanco. En ARASIS es posible conocer qué le ha ocurrido en cada momento de su ciclo de vida a cada objeto del sistema, y conocer cuándo, qué ha ocurrido y por quién. Es decir, tenemos los mecanismos para conocer en todo momento la historia y líneas de vida de los objetos: el *provenance* de los objetos.

Además la utilización del versionado de protocolos como estrategia ante la evolución de los protocolos, ha permitido mantener toda la información histórica sin pérdida alguna.

Los sistemas OcTrail y de Business Intelligence proporcionan unos entornos muy completos para la explotación total de toda la información almacenada en el biobanco. Esto permite a los investigadores del proyecto realizar análisis complejos de los datos para sus estudios en las enfermedades cardiovasculares, consultando en tiempo real todo el conocimiento almacenado en la base de acontecimientos.

Evaluación del rendimiento de ARASIS

La evaluación del rendimiento de ARASIS se ha realizado en base a la medición de dos dimensiones diferentes:

- La evolución del espacio de memoria utilizado por la base de datos a lo largo del tiempo.
- El tiempo de respuesta del sistema OcTrail de la herramienta en la obtención de varias líneas de vida.

Las pruebas de tiempo se realizaron con la herramienta desplegada en un servidor de aplicaciones Apache Tomcat instalado en un HP Compaq dc 7800 con un procesador Intel Core 2 Duo de 2500 MHz y 2 GB de RAM, mientras que para la base de datos se utilizó PostgreSQL como SGBD.

En primer lugar, el espacio de memoria se ha analizado midiendo el tamaño en KB que ocupan las copias de seguridad realizadas de la base de datos en diferentes momentos del tiempo, y

se ha comparado con el volumen de muestras y ejecuciones de protocolo existentes en ese momento para analizar su evolución. En la siguiente imagen se muestra esta gráfica comparativa:

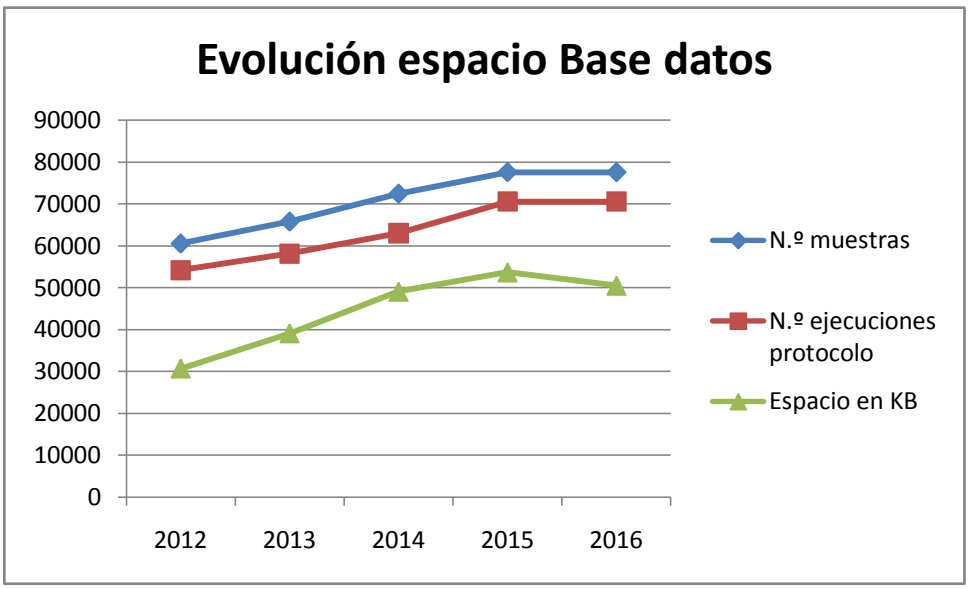


Figura 7-22 Evolución del espacio de almacenamiento

Como se puede observar, hasta el año 2015, el espacio utilizado por la base de datos crece linealmente y al mismo ritmo que el número de elementos utilizados en la comparativa, por lo que se podía prever el comportamiento de la herramienta con respecto al espacio necesario en el futuro.

En el año 2016 se reduce el espacio necesario a pesar de mantenerse más o menos constante el número de muestras y ejecuciones de protocolos. Esta reducción es debida al cambio del motor del SGBD PostgreSQL, pasando de la versión 8.3 a la 9.3, estando esta última más optimizada en cuanto al espacio necesario, por lo que incluso en un futuro se puede prever una necesidad menor conforme las nuevas versiones sean más optimizadas.

Por otro lado, el tamaño actual en MB de la copia de seguridad de la base de datos es de aproximadamente de 49,5 MB. Estas cantidades son insignificantes en los órdenes de Terabytes de almacenamiento en los que se mueven actualmente los sistemas de información.

En segundo lugar, la evaluación en cuanto al tiempo de respuesta consistió en medir el tiempo necesario para completar el procesamiento de las historias de varios empleados. Estos empleados se han elegido teniendo en cuenta que tuvieran un número diferente de muestras registradas para comparar cómo varía el tiempo en función del número de consultas a la base de acontecimientos a realizar. La elección de esta funcionalidad como prueba es debido a que la obtención de la historia de un objeto es la tarea más costosa y lenta que se realiza en un SGBAP, ya que implica la obtención de la línea de vida del objeto (todos los protocolos que han actuado sobre él con la información de todos los efectos que le provocan), así como de todos sus objetos relacionados.

En el caso especial de la obtención de la historia de un empleado, esta tarea además involucra

la obtención de líneas de vida de prácticamente todos los tipos de objetos de ARASIS, debido a las relaciones de dependencia existentes entre los objetos.

Los resultados obtenidos se muestran en la siguiente tabla:

Código empleado	N.º muestras	N.º objetos relacionados	N.º protocolos ejecutados	N.º consultas SQL	Tiempo (sg)
865393	1	2	11	16	0,431
570090	4	36	86	256	5,076
701941	8	58	136	403	5,910
781547	12	84	189	592	7,773
512200	16	138	370	989	15,394
929570	20	180	480	1321	19,678

Figura 7-23 Tiempos de respuesta para calcular la historia de varios empleados

En la tabla se muestra para cada empleado de la prueba, (1) el número de muestras que tiene registradas, (2) el número de objetos relacionados con esas muestras, (3) el número de protocolos ejecutados en el conjunto de su historia –a él directamente y todos sus objetos relacionados-, (4) el número de consultas SQL que ha sido necesario realizar contra la base de acontecimientos para obtener las líneas de vida, y finalmente (5) el tiempo de respuesta (en segundos). Hay que tener en cuenta que el tiempo de respuesta mide el proceso realizado en la capa de negocio en el servidor, sin incluir el tiempo necesario de representación en la interfaz, si bien es prácticamente residual.

Si analizamos los resultados podemos ver que los tiempos de respuesta son buenos, ya que, por ejemplo, para el caso del último empleado es un tiempo razonable si tenemos en cuenta que se están obteniendo 180 líneas de vida con la información de la ejecución de 480 protocolos para obtener la historia de ese empleado.

Además, hay que tener en cuenta que las condiciones de esta prueba pueden ser mejoradas, puesto que para el análisis de esta tesis las pruebas se han realizado desde un ordenador personal y no utilizando toda la potencia que ofrece, por ejemplo, un servidor profesional de aplicaciones donde la potencia de procesamiento es considerablemente superior.

En la siguiente imagen se representa el tiempo de respuesta con respecto al número de objetos implicados (es decir, el número de líneas de vida obtenidas), para analizar la evolución de los datos gráficamente, lo que nos dará una muestra del orden de tiempos de la obtención de la historia de un objeto.

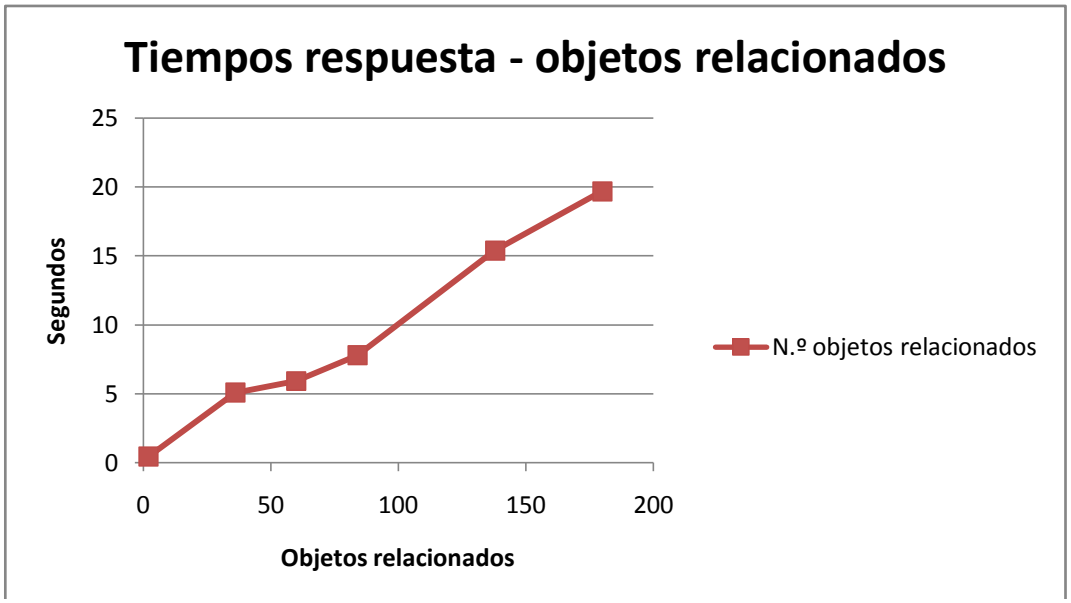


Figura 7-24 Evolución del tiempo de respuesta al calcular la historia

Como puede verse, existe una relación lineal entre las dos magnitudes, lo que nos permite concluir que el tiempo de respuesta es linealmente dependiente de la complejidad de la historia del objeto, lo que puede considerarse como un buen resultado de rendimiento.

Capítulo 8. Una estrategia alternativa: optimizando el acceso al tiempo

En el Capítulo 5 se ha detallado una estrategia de diseño para construir sistemas de monitorización utilizando el concepto de acontecimiento, los sistemas de gestión de bases de acontecimientos basados en protocolos. Esta estrategia, como hemos visto en el apartado anterior en la evaluación de su aplicación en ARASIS, nos permite construir sistemas que gestionan y explotan de forma eficiente toda la información que se almacena en la base de acontecimientos. Aun así, la obtención de unos buenos resultados no es óbice para que se siga investigando para mejorar y optimizar las estructuras que se han desarrollado y, en este caso, probado con éxito.

En este capítulo, se presenta una alternativa en la estrategia de diseño que se ha elaborado con la intención de mejorar el proceso de obtención de las líneas de vida de los objetos, optimizando las estructuras definidas en los patrones de acontecimientos. Esta modificación consiste en variar el diseño realizado para representar el efecto producido por la ejecución de un protocolo, respetando evidentemente su definición dentro del concepto de acontecimiento.

En la estrategia de diseño anterior, la ejecución de un protocolo estaba asociada con la transición de cambio de estado que se producía como efecto de su ejecución. Sin embargo, esta representación tiene el inconveniente de que para conocer la información del protocolo que ha generado un estado ha de consultarse específicamente dicha transición, aumentando el número de consultas a realizar para ello y, por tanto, perdiendo eficiencia.

Para evitar este inconveniente, la variante que se propone en esta alternativa es que la asociación de la ejecución del protocolo se realice directamente con el estado al cual cambia el objeto. Conceptualmente esta variación es correcta, ya que el resultado del efecto de una ejecución es realmente el estado al que cambia el objeto, siendo la transición el medio por el cual el objeto ha llegado a ese estado. Ambos conceptos, la transición entre estados y el estado que se genera son parte del efecto del protocolo en el acontecimiento, por lo que ambas representaciones son válidas.

Por otro lado, en esta alternativa también se modifica el lugar donde se almacena el momento en el cual el objeto ha cambiado al estado en la ejecución del protocolo. En la estrategia anterior, este instante se registra en la acción que realiza la ejecución del protocolo sobre el objeto, definida en la asociación de la ejecución con la transición entre estados, representando el momento en el cual la acción provoca el cambio de estado. En esta alternativa, este valor de tiempo se modela en el propio estado efecto de la ejecución del protocolo, almacenando el instante en el que se ha generado el estado del objeto. Además, al hacer este cambio la acción se puede transformar de clase asociativa a asociación directamente, ya que no se requiere de una clase para contener la información del tiempo.

En los dos casos se representa realmente el mismo instante dentro del acontecimiento, ya que el instante en el que se ejecuta la acción es el instante en el que se genera el estado; sin embargo, lo que más interesa conocer conceptualmente es el instante en el que el objeto comienza a estar en un determinado estado, y en este diseño está directamente modelado en la propia entidad en la que se necesita, sin tener que ser resultado de su deducción por el tiempo de ejecución de la acción.

En la siguiente imagen se muestra la variación que se propone utilizando el patrón de acontecimientos de cambio de estado.

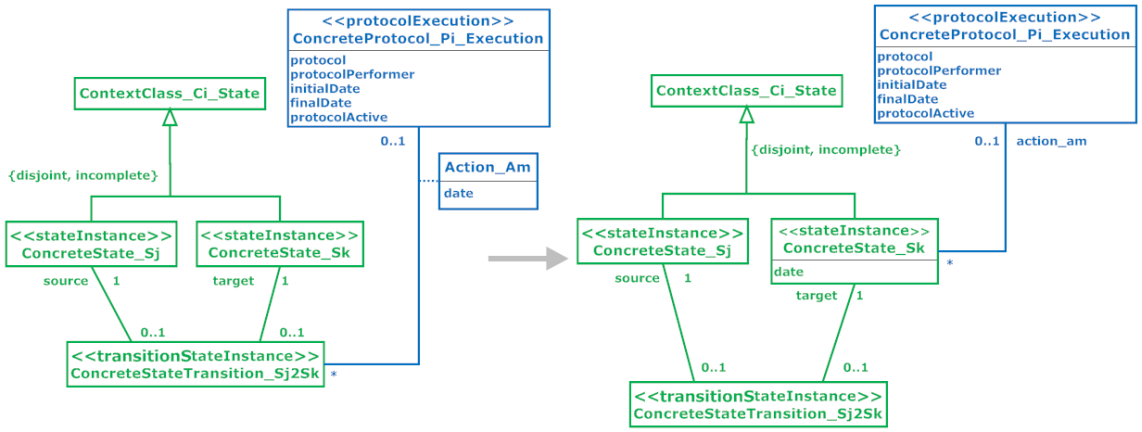


Figura 8-1 Modificación en la estrategia de diseño

Con el objetivo de observar con mayor claridad el cambio, la imagen se restringe únicamente al cambio de la asociación de la ejecución del protocolo. Como se puede observar, con este nuevo diseño se mejora la obtención tanto de la información del protocolo que genera el estado, como el instante en el que el estado se ha generado, lo que se traduce en una consecuente optimización de la obtención de la información de un acontecimiento.

Estas modificaciones se han realizado también en los patrones de acontecimientos de datos, ya que en estos casos nos encontramos con las mismas problemáticas de acceso al protocolo y tiempo en el que se generan los acontecimientos.

8.1 Patrones de acontecimientos

En las siguientes imágenes se muestra cómo quedan los patrones de acontecimientos con los cambios descritos en la introducción del capítulo.

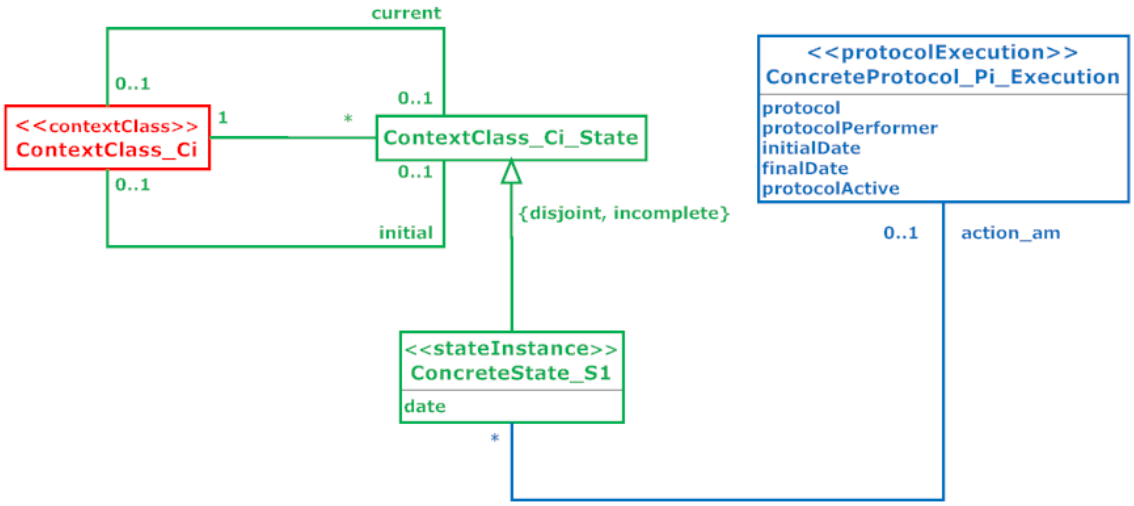


Figura 8-2 Patrón de acontecimientos de creación de objetos en la nueva estrategia

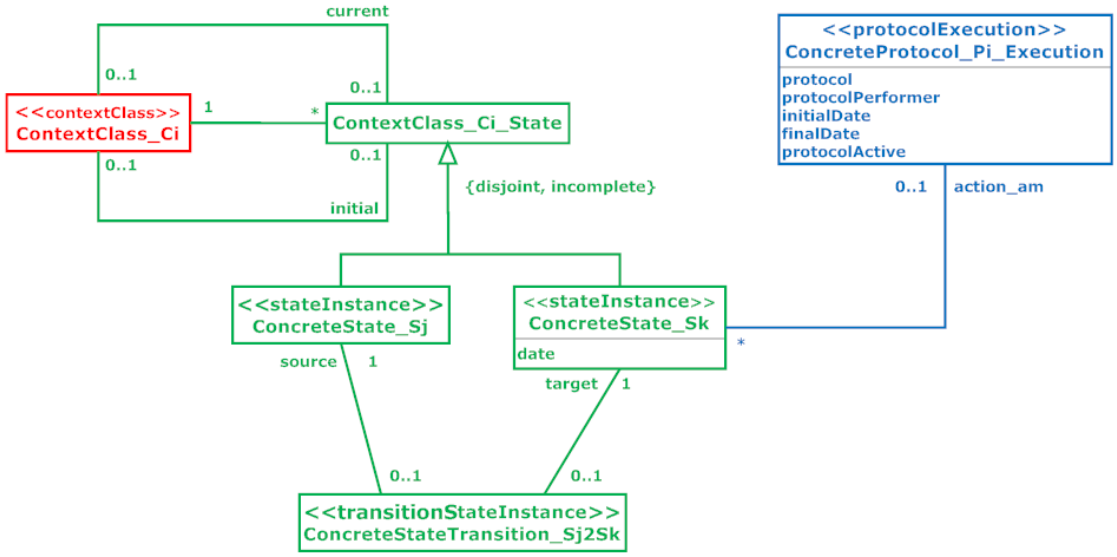


Figura 8-3 Patrón de acontecimientos de cambio de estado en la nueva estrategia

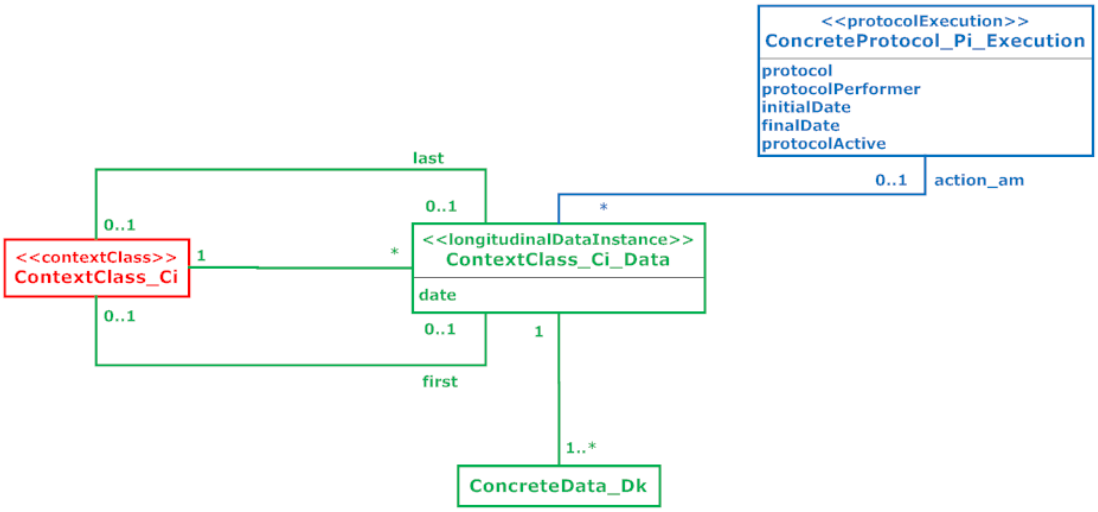


Figura 8-4 Patrón de acontecimientos de registro de datos de objeto en la nueva estrategia

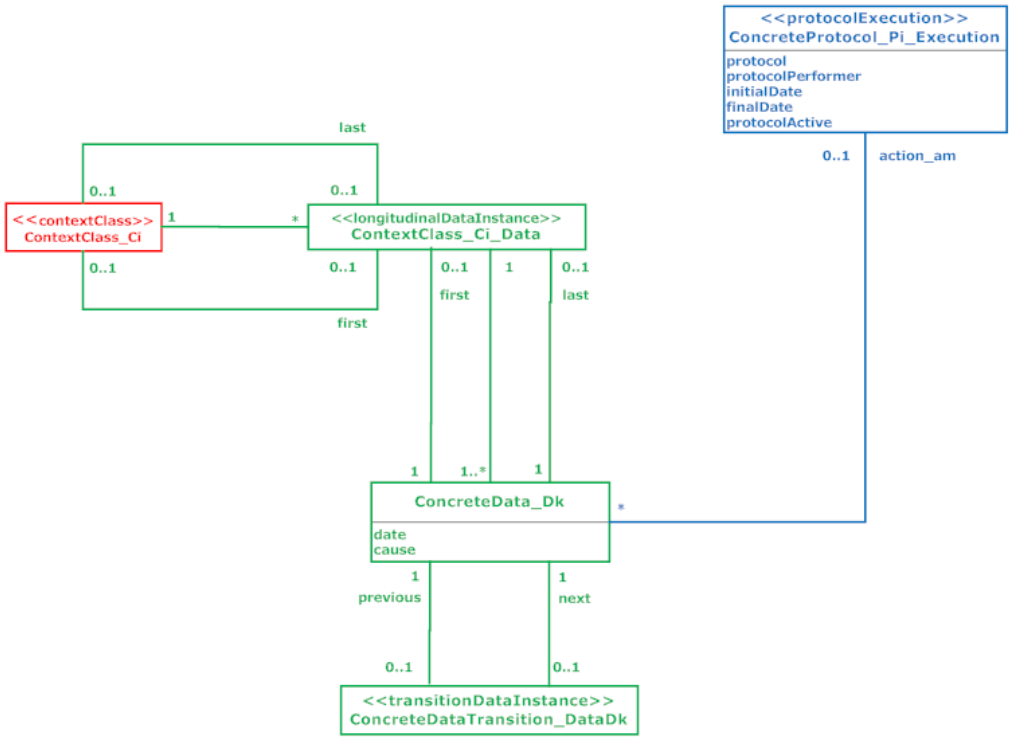


Figura 8-5 Patrón de acontecimientos de cambio datos de objeto en la nueva estrategia

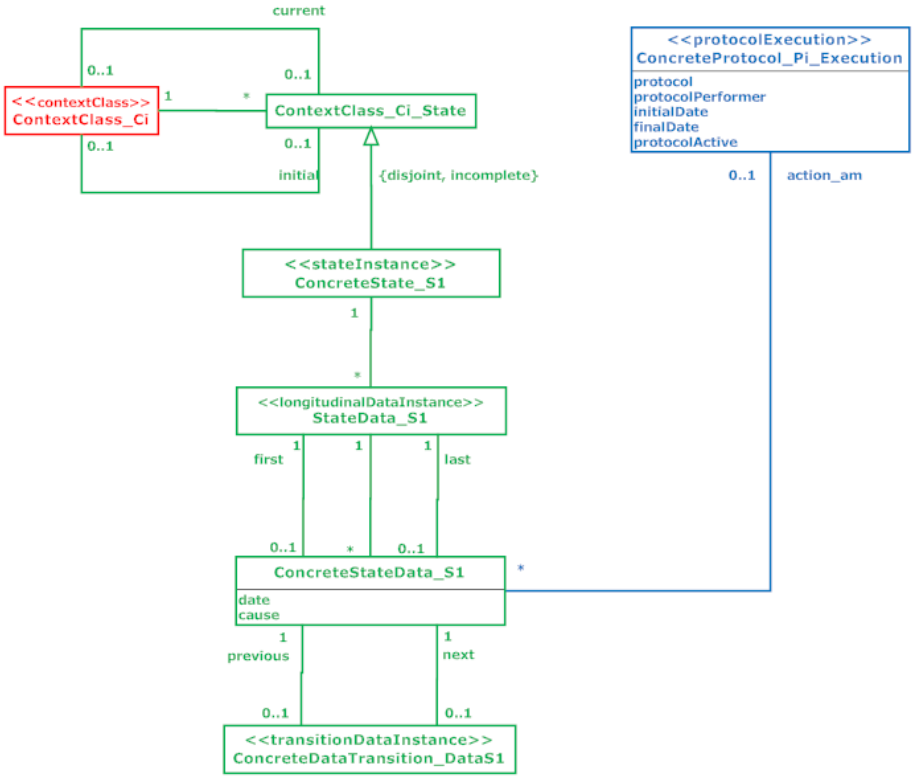


Figura 8-6 Patrón de acontecimientos de cambio datos de estado en la nueva estrategia

Si analizamos los patrones anteriores, se puede observar que se homogeniza la obtención de la información de los acontecimientos. Ahora, en todos los patrones la ejecución de los protocolos se asocia directamente con el estado o los datos longitudinales. Esto evidentemente provoca una mejora en la eficiencia de la obtención de los acontecimientos de la línea de vida de un objeto, ya que no se requiere de una particularización de la forma de obtener la información según sea el caso.

En los casos de los patrones de creación de objeto y de registro de datos, como el protocolo ya se asociaba directamente con el estado o dato longitudinal respectivamente, el cambio sólo consiste en la localización del atributo que almacena el tiempo, pasando de la acción al estado o dato.

Por último, la modificación de los patrones de acontecimientos implica que se modifiquen los patrones PIM y PSM definidos en la metodología de diseño. En las siguientes imágenes, puede verse el resultado de estas modificaciones:

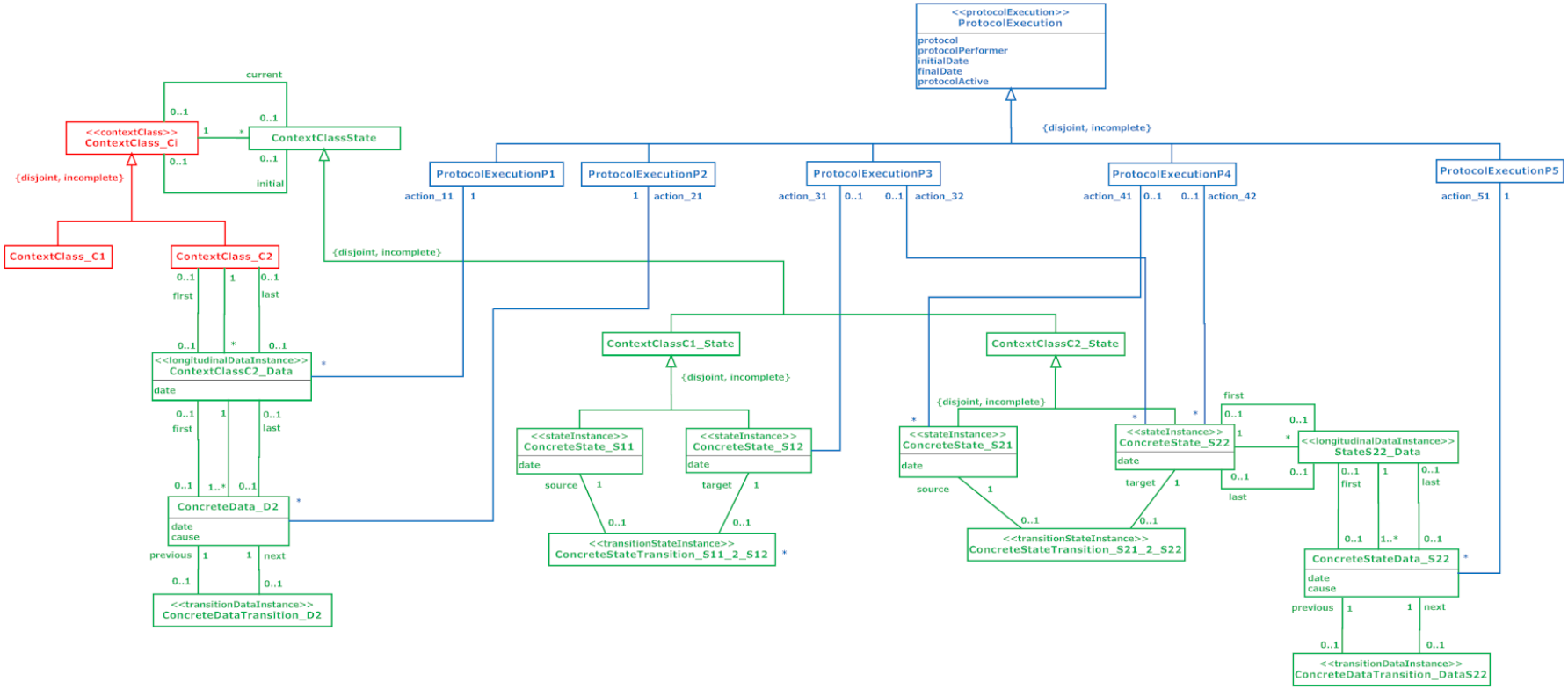


Figura 8-7 Patrón PIM en la nueva estrategia

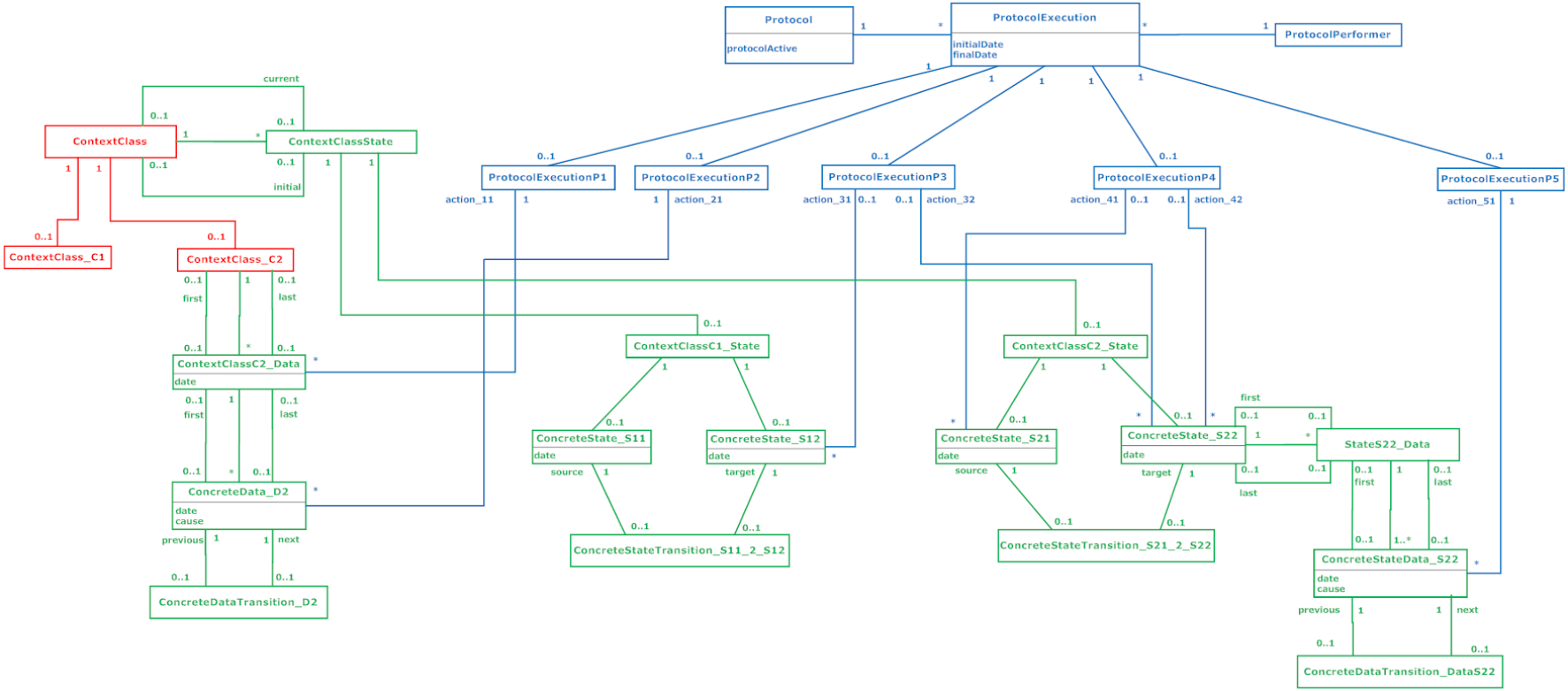


Figura 8-8 Patrón PSM en la nueva estrategia

El desarrollo de esta alternativa, como se ha descrito, afecta a la metodología de diseño de las bases de acontecimientos, permaneciendo inalterada la arquitectura del sistema propuesta en la estrategia de diseño.

8.2 Aplicación al caso ARASIS

La alternativa que hemos propuesto en este capítulo es el resultado de las últimas investigaciones realizadas sobre la estrategia de diseño, y el objetivo es su aplicación progresiva en los sistemas SGBAP que están actualmente construidos, como es el caso de ARASIS, siendo por su puesto la estrategia de diseño que se utilizará para los nuevos sistemas que se vayan desarrollando.

Respecto a su aplicación a ARASIS, dada la envergadura del sistema, actualmente se está en la fase de modificación de los modelos conceptuales. En las siguientes imágenes se puede ver el cambio que se produce en los patrones de acontecimientos de los ejemplos mostrados en la estrategia anterior. Se muestran tanto el diseño original como el diseño con la nueva estrategia.

Finalmente, se muestran los extractos de los modelos PIM y PSM de ARASIS modificados según la nueva estrategia de diseño.

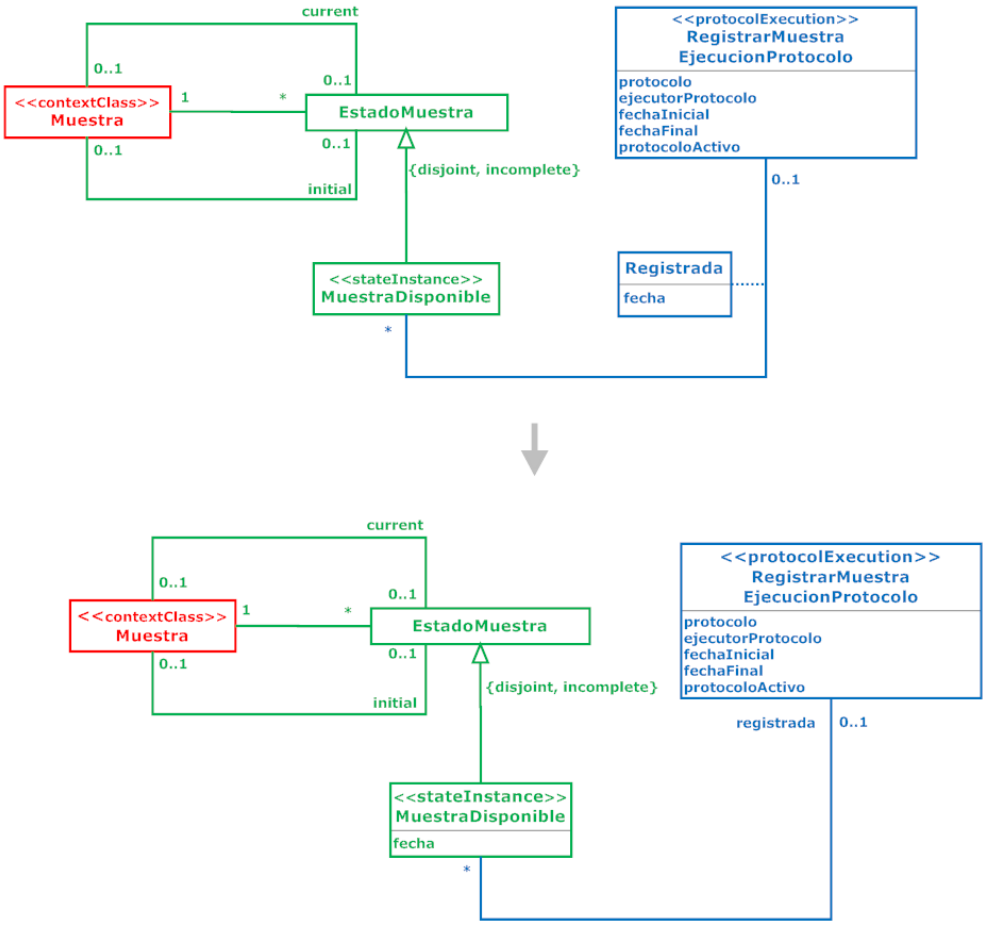


Figura 8-9 Ejemplo de aplicación del patrón de acontecimientos de creación de objeto en ARASIS según la nueva estrategia

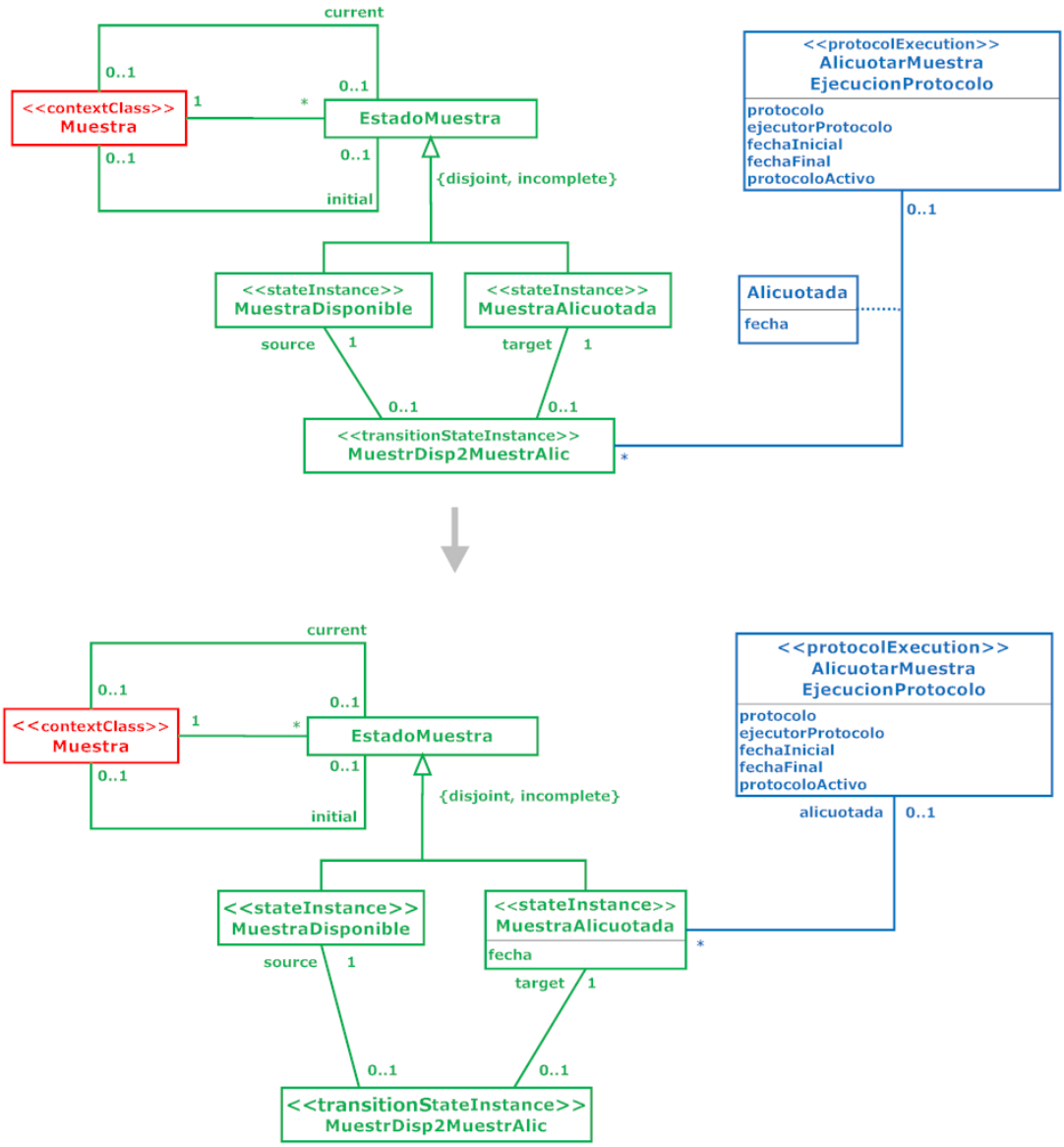


Figura 8-10 Ejemplo de aplicación del patrón de acontecimientos de cambio de estado en ARASIS según la nueva estrategia

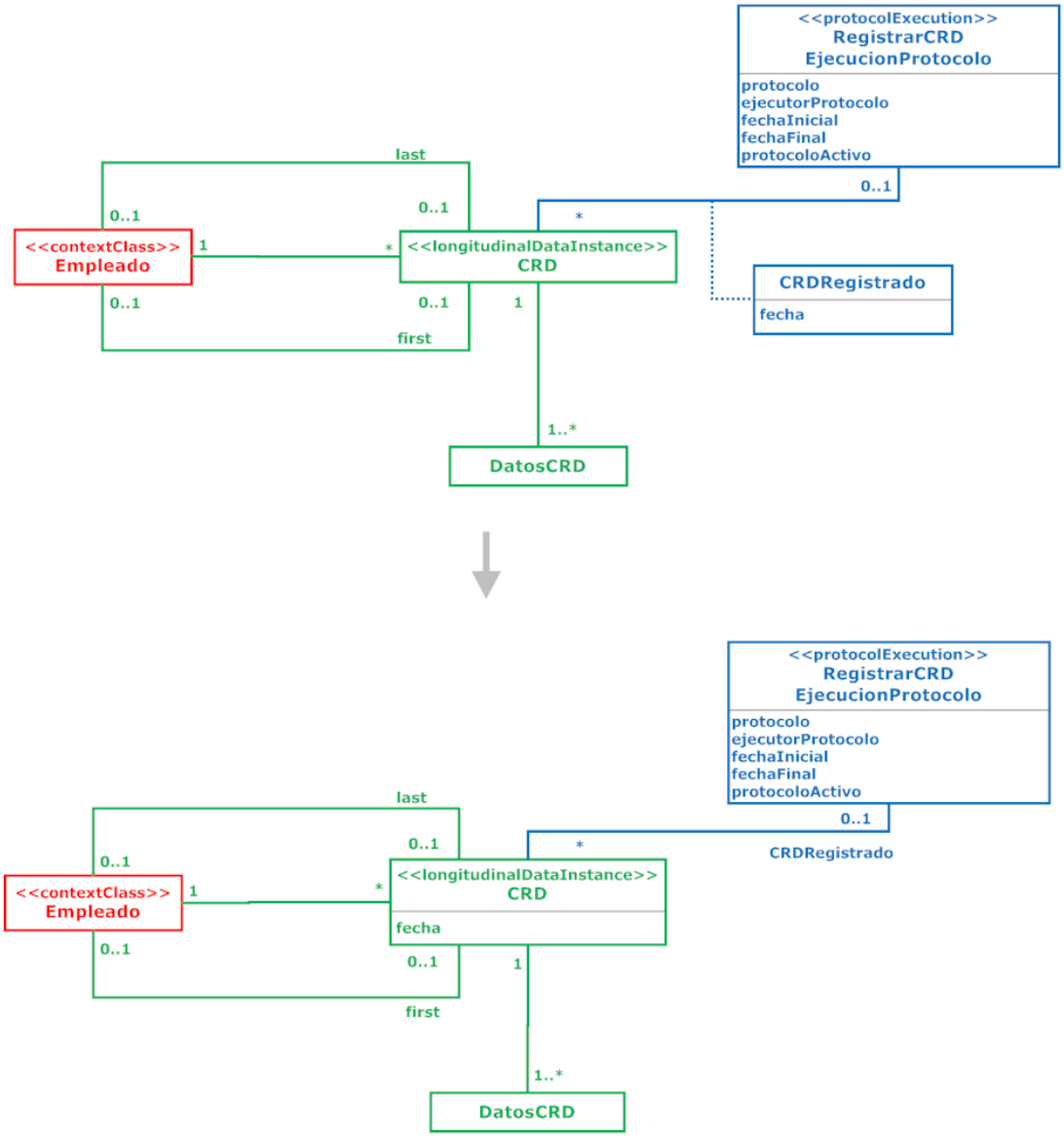


Figura 8-11 Ejemplo de aplicación del patrón de acontecimientos de registro de datos de objeto en ARASIS según la nueva estrategia

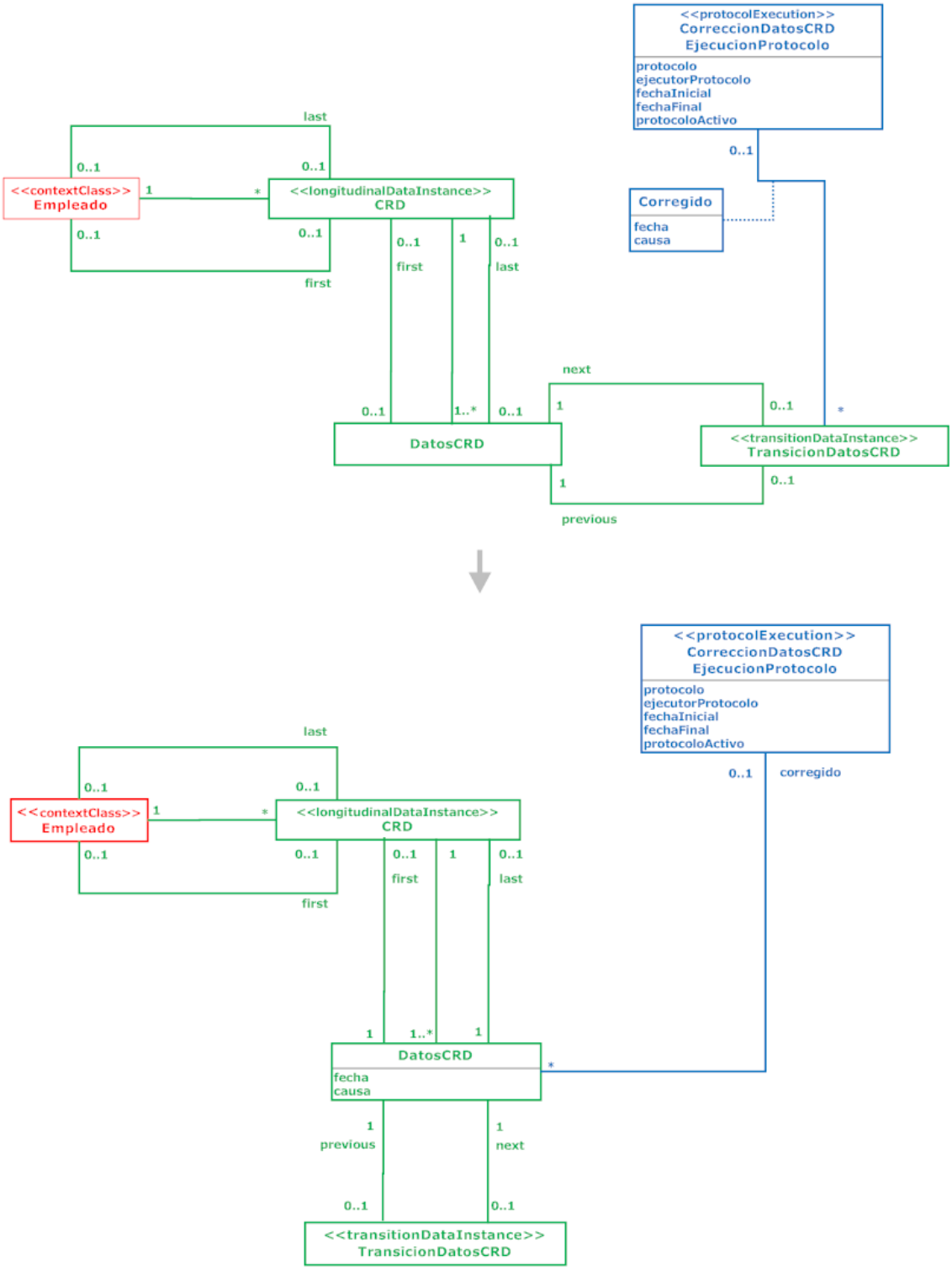


Figura 8-12 Ejemplo de aplicación del patrón de acontecimientos de cambio datos de objeto en ARASIS según la nueva estrategia

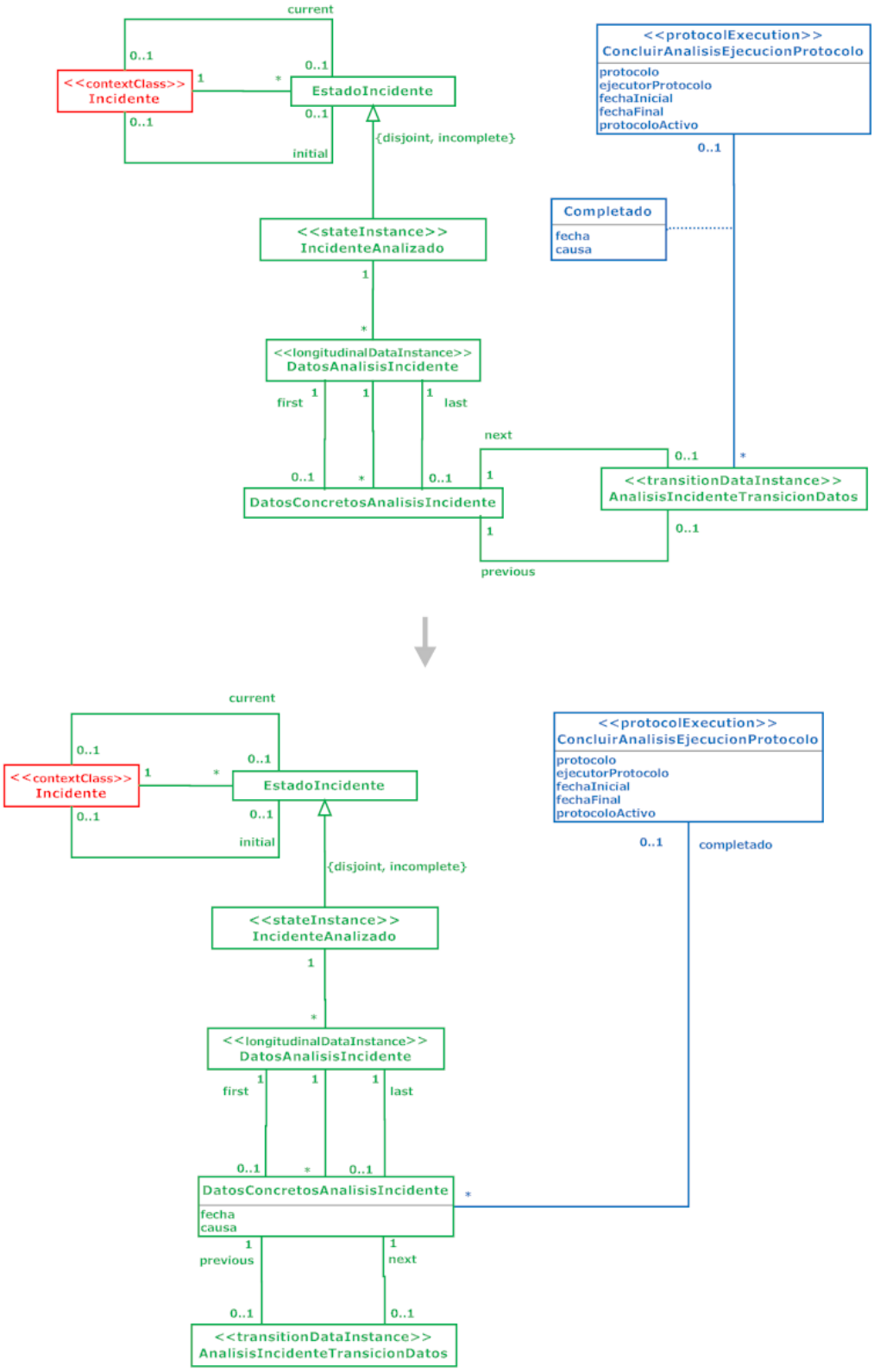


Figura 8-13 Ejemplo de aplicación del patrón de acontecimientos de cambio datos de estado según la nueva estrategia

En este último caso, el ejemplo corresponde al análisis de un incidente ya que en ARASIS no se ha utilizado el patrón de acontecimientos de cambio de datos de estado.

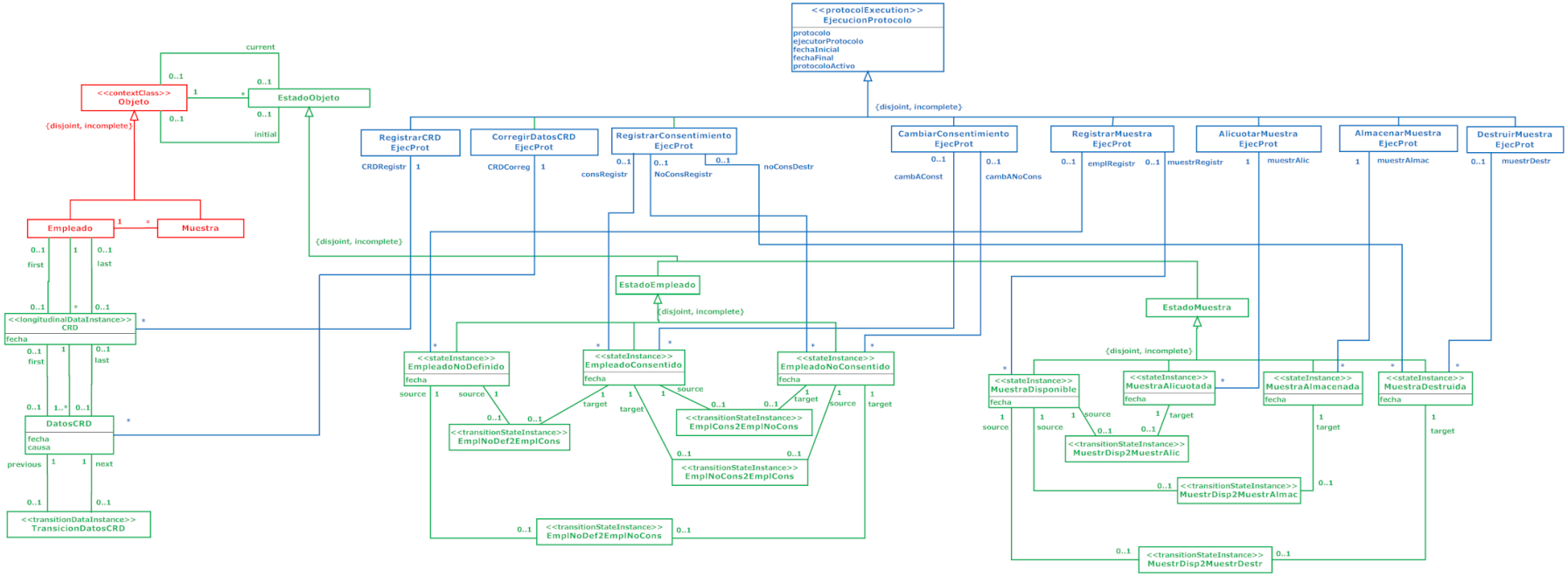


Figura 8-14 Extracto del modelo PIM de ARASIS según la nueva estrategia

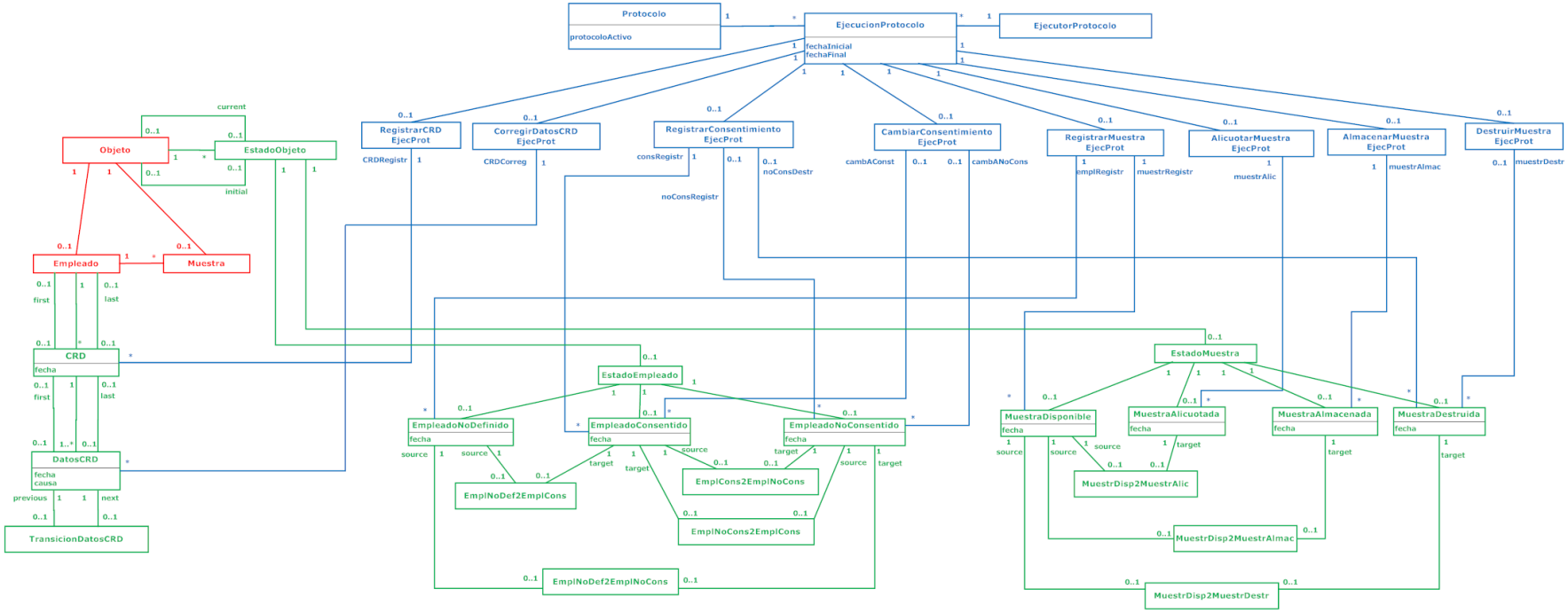


Figura 8-15 Extracto del modelo PSM en ARASIS según la nueva estrategia

IV. ACONTECIMIENTOS BASADOS EN TAREAS

Capítulo 9. Gestión de tareas: Acontecimientos basado en tareas

Las empresas se esfuerzan por sobrevivir y progresar en entornos cada vez más exigentes, por lo que compararse con el mercado es vital para definir su posición con respecto a sus competidores, y en qué grado van a cumplir o excederse en las necesidades y expectativas de sus socios, clientes y usuarios. Por ello, tener una foto realista, precisa y actualizada de su nivel de desempeño es un requisito básico para cualquier iniciativa de mejora, además de ser un valioso apoyo en la toma de decisiones estratégicas futuras.

Monitorizar y evaluar el desempeño individual de cada uno de los recursos de un sistema ayuda a detectar y corregir posibles desviaciones sobre el nivel de calidad esperado, así como asegurar el cumplimiento de los resultados fijados. Esta evaluación del desempeño se realiza mediante indicadores diseñados específicamente que relacionan la calidad del resultado de la tarea realizada y el coste (cómo de eficiente se han utilizado los recursos, cuánto tiempo ha costado, etc.).

Por otro lado, esta monitorización proporciona como valor añadido un conocimiento actualizado y preciso de los recursos disponibles y sus capacidades. Podemos conocer qué es capaz de hacer cada recurso, cómo de bien puede hacerlo y a qué coste. Este conocimiento es un input muy importante en la planificación del negocio, ya que ayuda a que cada recurso sea asignado a las tareas para las que es más adecuado.

La gestión de tareas en el mundo empresarial es una disciplina compleja que tiene como uno de sus objetivos la optimización de la asignación de las tareas que tienen que realizar los recursos de la empresa [59]. Para ello, se utilizan diversas técnicas que permiten efectuar esta asignación mediante métodos y algoritmos que intentan minimizar el coste que supondría que los recursos realizaran las tareas, en términos tanto económicos como de tiempo. Sin embargo, estos métodos de asignación normalmente no consideran un aspecto fundamental en el funcionamiento de un sistema o empresa, la historia de cómo han desempeñado los recursos las tareas en el pasado.

En este capítulo, veremos la aplicación del modelo de acontecimientos a la gestión de tareas. En concreto, se propone un **patrón** de diseño de estructuras de persistencia **para la evaluación del rendimiento** de los recursos a través de la historia de acontecimientos registrados, el ***Patrón de Acontecimientos basados en Tareas***. La evaluación se realizará mediante la definición de **indicadores** sobre los acontecimientos. Los acontecimientos e indicadores nos permitirán, además, precisar la información que utilizan de entrada los métodos de asignación de tareas en base a la historia del desempeño, obteniendo como resultado una asignación de tareas más eficiente en nuestros sistemas.

Con este objetivo, como elemento previo al *Patrón de Acontecimientos basados en Tareas*, se

define una variante del concepto de acontecimientos llamado ***acontecimiento basado en tareas***. En esta variante, el objetivo es registrar la realización de una tarea por un actor de un entorno o sistema, siendo el actor el núcleo de la información para poder evaluar su rendimiento. Este enfoque difiere del acontecimiento basado en protocolos, donde la estructura estaba centrada en el objeto del sistema sobre el que actuaba el protocolo, para conocer qué efecto había provocado el protocolo en dicho objeto.

En el patrón se diseña la estructura de persistencia para almacenar los acontecimientos basados en tareas. Esta estructura nos va a permitir realizar una evaluación completa y eficaz del desempeño de un objeto utilizando para ello, junto al conocimiento de las capacidades teóricas de los recursos, la historia de las tareas que ha realizado en el tiempo.

El concepto de *capacidad* de un recurso puede ser entendido como la habilidad de éste para realizar una actividad. Su significado preciso depende del entorno en el que se considere. Por ejemplo, consideremos tres contextos diferentes:

- En el primer caso, consideremos una empresa compleja, la cual está compuesta por unidades de negocio, como el servicio de atención al cliente.

  Las capacidades de estas unidades se basan en el conocimiento y experiencia obtenidos en el tiempo, lo que les ha permito construir y definir unos procesos de negocio bien diseñados.

  En el caso de la unidad de Atención al Cliente, su capacidad puede ser evaluada mediante indicadores relacionados con la efectividad y eficiencia en la gestión de las peticiones e incidencias.
- En el segundo caso consideremos el sistema de salud que, entre otros recursos, está compuesto por personas, como los médicos.

  La capacidad de un médico resulta de su entrenamiento, conocimiento, experiencia y habilidades. Puede ser medida, por ejemplo, por la precisión de sus diagnósticos y por el uso adecuado de los recursos disponibles, como el de los materiales y equipos utilizados y las horas de personal invertidas en la realización de pruebas diagnósticas.
- El tercer escenario es un sistema de vigilancia con varios robots.

  Su éxito depende de una eficiente asignación de tareas, basada en el conocimiento de lo que son capaces de hacer sus componentes internos y cómo de bien lo han realizado en la práctica.

  En este caso, la capacidad refiere al uso eficiente y efectivo de las capacidades de sus componentes (motor, sensores, comunicación, memoria y procesamiento), comprobados en situaciones reales.

Como prueba de concepto, se desarrollará la aplicación del patrón en cada uno de los tres contextos anteriores.

La descripción de los conceptos del Patrón de Acontecimientos basados en Tareas se ha publicado en el artículo [3], artículo presentado en el congreso EuroPloP del año 2016 sobre patrones y lenguajes de patrones [32].

9.1 Asignación de tareas

El concepto de asignación de tareas –task allocation- se utiliza en muchos contextos del mundo real con propósitos diferentes, como en la economía, en la teoría de juegos, en la investigación operativa o en la asignación de personal [89]. También se utiliza dentro de algunas de las líneas de investigación en auge ahora mismo, como la asignación de tareas en entornos distribuidos de sistemas o máquinas para conseguir un objetivo común mediante métodos de inteligencia de enjambre [12].

Por asignación de tareas entendemos el proceso por el cual se asigna la realización de una tarea a un recurso concreto de la manera más óptima posible [37]. Un ejemplo típico de este proceso es la asignación de las tareas a realizar por un conjunto de máquinas, en la que cada máquina incurre en un coste determinado cuando realiza cada tarea.

El proceso de asignación de tareas se puede dividir en tres fases:

- Identificación, categorización, análisis y desglose en actividades atómicas de las tareas, definiendo los criterios de éxito (por ejemplo, la calidad deseada del resultado) y las restricciones existentes (costes, tiempos, políticas, etc.).
- Identificación y análisis de qué es capaz de realizar cada recurso, y a qué coste, para obtener un conocimiento actualizado y preciso de los recursos disponibles.
- Ejecutar un método para asignar de forma óptima las tareas a los recursos. Idealmente, el método maximizará los beneficios a un coste óptimo.

El patrón para la evaluación que se define en este capítulo se focaliza en la segunda fase, proporcionando un conocimiento completo, preciso y actualizado de los recursos; conocimiento que podrá ser utilizado por los métodos que resuelven los problemas de asignación de tareas para una asignación más óptima.

A este tipo de problemas se les llama *problema de asignación* [17], y es uno de los problemas existentes en la rama de la optimización combinatoria en la investigación operativa [42]. En el problema de asignación tradicional cada tarea se asigna a un recurso y un recurso sólo puede asignarse a una tarea. Sin embargo, el problema de asignación se ha generalizado para contemplar muchas otras opciones en cuanto a simultaneidad de realización de tareas, trabajos colaborativos de recursos para resolver una tarea y otro tipo de particularidades.

Un claro ejemplo de estas líneas son las investigaciones que se realizan en el contexto de sistemas multiagente o multirobot. En [37], se dispone de una taxonomía de asignación de tareas en este tipo de contextos, categorizando las soluciones en función de la capacidad de un robot de realizar una única –ST- o múltiples tareas –MT-, y el requerimiento de las tareas de ser realizadas por un único robot –SR- o por varios robots –MR-.

La mayoría de las propuestas de asignación de tareas en la literatura se alinean con el problema tradicional, perteneciendo a la categoría ST-SR (Single Task, Single Robot), en donde un robot ejecuta una tarea y cada tarea es ejecutada por un único robot [38][55]. En el lado opuesto, se encuentran los problemas MT-MR (Multi Task, Multi Robot) [85]: un robot puede ejecutar múltiples tareas simultáneamente, y algunas tareas requieren de múltiples robots.

La asignación de tareas en este tipo de problemas puede reducirse a un problema de optimización. La función a optimizar, llamada utilidad en [37], resulta de una confluencia

entre las capacidades y costes de los recursos. La mayoría de estas funciones son 'negativas', es decir, miden algún tipo de coste que debe ser minimizado, como la distancia media para completar las tareas, el tiempo de completar las tareas, o el uso de recursos que se necesitan para ello [38][58][5]. En el caso de funciones 'positivas', lo que intentan es maximizar un beneficio que se determine en el sistema, como una tasa de aciertos/fallos que se disponga de los recursos en la realización de las tareas [51].

Como problemas de optimización que son, las técnicas de resolución más habituales provienen del campo de la investigación operativa, siendo generalmente las soluciones de programación lineal y metaheurísticas las más utilizadas [73]. En estas técnicas, el objetivo es maximizar o minimizar, dependiendo de si las variables a optimizar son costes o beneficios, una función de utilidad.

Dentro del estudio del problema tradicional de asignación, también se hace referencia al método Húngaro, como método tradicional de resolución del problema. Este método es un algoritmo de optimización que resuelve problemas de asignación mediante la aplicación de transformaciones en una serie de pasos. Fue publicado por Harold W. Kuhn en 1955, y revisado por James Munkres en 1957, siendo conocido, además de método o algoritmo húngaro, como algoritmo de la asignación de Munkres, o el algoritmo de Kuhn-Munkres [65].

9.2 El uso de la historia para determinar la asignación de tareas

Un problema de las técnicas de resolución del problema de asignación de tareas, es que habitualmente se realiza sobre una función de utilidad que considera una variable, como el coste, en la que sus valores provienen de referencias o estadísticas fijas sin considerar su evolución a lo largo del tiempo. Sin embargo, en ciertas ocasiones se necesita maximizar la calidad de la ejecución de la tarea asignándosela al recurso más capaz de realizarla en base a la calidad de su desempeño en el pasado.

En este contexto, conocer la historia de las tareas que ha realizado cada recurso, y cómo las ha realizado, es una información importantísima que nos va a permitir realizar la asignación de tareas más adecuada.

Por ejemplo, imaginemos una empresa que necesita resolver urgentemente una incidencia grave que se ha producido en su sistema, y que para esa tarea disponen de la capacitación suficiente tres recursos –equipos de trabajo-: *e1*, *e2* y *e3*.

En términos de coste de ejecución, podemos tener que el equipo *e2* es el más económico y el equipo *e3* el más costoso. Sin embargo, dada la urgencia de la incidencia, necesitamos asignar su desarrollo al equipo que antes la resuelva y con una tasa de éxito (que mide el porcentaje de veces que el equipo ha resuelto una incidencia de esas características con éxito) de más del 90%, dejando un poco al margen ese factor económico.

Para poder realizar la asignación de forma adecuada en estas condiciones, es imprescindible conocer la historia de cómo han resuelto estos tres equipos otras incidencias en el pasado. Por ejemplo, supongamos que tenemos registrado que el equipo *e1* tarda de media 2 días en la resolución de incidencias similares, y con una tasa de éxito del 92 %. El equipo *e2*, tarda también 2 días de media y con una tasa de éxito del 85%, y el equipo *e3* tarda 3 días de media y con una tasa de éxito del 96%.

Por tanto, en esta situación, aunque el equipo *e2* nos resolvería la incidencia en el mismo tiempo que el equipo *e1* (siendo más económico en coste), la historia de su comportamiento en el pasado nos indica que su tasa de éxito es menor al nivel que necesitamos, por lo que la tarea se asignaría al equipo *e1*. El equipo *e3*, si bien tiene la mayor tasa de éxito de todos, tarda más tiempo de media en realizar la tarea.

En la literatura existen algunos modelos de asignación de tareas utilizando información histórica de comportamientos pasados. En [81], por ejemplo, se propone un método de búsqueda de información de asignaciones de tareas pasadas mediante técnicas de data mining. En este método, se analiza la información de ejecuciones de procesos antiguas y la información del contexto de las tareas a asignar, determinando un modelo mediante árboles de decisión en función de los análisis de información realizados. En la solución propuesta en [51], la probabilidad de que un robot realice una tarea depende de su historia de éxitos / fracasos en la realización de las tareas. Sin embargo, en esta propuesta los robots únicamente están capacitados para realizar una única tarea.

En cualquier caso, para una asignación de tareas óptima, se requiere un modelo que facilite tener registrada la información histórica de la ejecución de tareas por parte de los recursos del sistema, y que posibilite:

- Conocer qué tareas ha realizado cada recurso en el transcurso del tiempo.
- Conocer cómo han ejecutado esas tareas, con información temporal de cuándo y durante cuánto tiempo han realizado las tareas, y con información acerca de los resultados de dichas ejecuciones.
- Definir medidas sobre ese conocimiento que permitan evaluar el desempeño de los recursos.
- Ofrecer una solución adaptable a cualquier tipo de escenario de aplicación, tanto en asignaciones de una tarea a un recurso, como de múltiples tareas a múltiples recursos.

Así, podremos desarrollar mecanismos que permitan almacenar esta información y utilizarla para realizar una asignación óptima de tareas utilizando la historia. Uno de estos mecanismos es el patrón para la evaluación del rendimiento basada en la historia que se desarrolla en el apartado 9.4.

El objetivo del patrón es almacenar la información de la ejecución de las tareas por parte de los recursos del sistema, en forma de acontecimientos, para poder establecer indicadores que permitan evaluar su desempeño. Este conocimiento nos permitirá mejorar los datos de entrada de los métodos de resolución de los problemas de asignación existentes actualmente, logrando con ello optimizar la asignación de tareas con la historia almacenada de los recursos.

En el siguiente apartado se define el concepto de *acontecimiento basado en tareas*, como variante del concepto general de acontecimiento, que establece los fundamentos teóricos que se utilizan en el desarrollo del patrón de evaluación del rendimiento.

9.3 La noción de acontecimiento basado en tareas

Si recordamos el concepto general de acontecimiento, definido en el apartado 3.1, se definía como la unidad mínima de información en una base de acontecimientos de acuerdo a tres

dimensiones: la guía, la estructura y el comportamiento.

En el caso del acontecimiento basado en protocolos (apartado 4.2), la guía son los protocolos que actúan sobre unos objetos –estructura-, provocándoles un cambio de estado o de los datos almacenados sobre ellos –comportamiento-. En este caso, por tanto, el foco se centra en los objetos sobre los que se ejecutan los protocolos, ya que el objetivo es registrar qué efecto les producen dichas ejecuciones.

En el contexto de la realización de tareas, podemos identificar igualmente los elementos que componen la definición tridimensional del acontecimiento.

Por un lado, están los recursos o unidades funcionales que realizan las tareas, que serán los objetos a considerar en el sistema –estructura-. Es decir, en este caso el foco principal, el núcleo, sobre los que se recoge la información son los ejecutores de las tareas, ya que el objetivo final del patrón es evaluar su desempeño en el sistema.

Los recursos tienen unas capacidades que les permiten realizar una serie de tareas en el sistema –guía-, y cuyas ejecuciones provocarán un efecto en forma de cambios de estado en dichos recursos–comportamiento-.

Por ejemplo, imaginemos el caso de un robot en un sistema de vigilancia en el que una de sus tareas sea vigilar un espacio. La realización de esa tarea conllevará que el robot cambie su estado, por ejemplo, de un estado 'en inactividad' a un estado 'ocupado' mientras está vigilando y de nuevo al estado 'en inactividad' una vez haya concluido. En este caso, la ejecución de la tarea produciría dos acontecimientos, correspondientes a ambos cambios de estado del robot.

De esta forma, podemos extender la definición de acontecimiento para definir el acontecimiento basado en tareas:

> Un ***acontecimiento basado en tareas*** es una pieza de información concreta, identificable e indivisible que contiene aspectos organizados de acuerdo a tres dimensiones -guía, estructura y comportamiento- proporcionados por la ejecución de una tarea, un objeto y un efecto.

Es importante destacar que, al igual que el acontecimiento general, el acontecimiento basado en tareas es la unidad mínima de información debido a que:

- Es único por naturaleza, ya que entre los elementos que lo definen está la realización de una tarea, que es única cada vez que ocurre en el tiempo, siendo por tanto identificable sin ambigüedades.
- Cuando se identifica un acontecimiento, cada uno de los elementos que lo componen no son significativos por sí mismos.

Representación de un acontecimiento basado en tareas

Si recordamos, el concepto de acontecimiento, de forma general, se representaba mediante una tupla de tres elementos:

(a, o, e)

donde *a* era la representación de la acción, el acontecer, que actuaba sobre o por acción de un objeto *o,* provocándole el efecto *e*.

En el caso del acontecimiento basado en tareas, su representación equivalente sería:

(t, o, d)

donde *t* es la tarea, el acontecer, que es realizada por un objeto *o,* dando lugar a los datos *d* asociados al cambio en el objeto en un momento determinado del tiempo (el efecto).

La componente *d* se refiere a cualquier cambio que le suceda al objeto al realizar la tarea, ya sea cualquier cambio de estado o información del resultado de realizar la tarea, sin un formato o estructura fija para que esta notación sea genérica independientemente del sistema.

Por ejemplo, en el ejemplo anterior del robot en el sistema de vigilancia, la realización de la tarea de vigilar el espacio generaba dos acontecimientos, que se pueden representar del siguiente modo:

(iniciarTareaVigilar001, robot01, (ocupado01, '2017-01-25 12:15'))

(finalizarTareaVigilar001, robot01, (inactivo01, '2017-01-25 13:15'))

9.4 Patrón para la evaluación del rendimiento basada en la historia

En este apartado se especifica el Patrón de Acontecimientos basados en Tareas, en el que se diseña, a partir de la noción abstracta de acontecimiento basado en tareas, la estructura de persistencia que nos permitirá almacenar los acontecimientos que ocurran en la realización de las tareas. En este patrón, a los recursos o unidades funcionales, es decir, a los objetos en el acontecimiento basado en tareas, los llamaremos *entidades* para utilizar una notación más general.

Para describir el patrón se va a utilizar el lenguaje desarrollado en el ámbito de la arquitectura por Alexander [2], definiendo: el *contexto* en el que la solución es aplicable, el *problema* que se necesita resolver, la *solución* y sus *consecuencias*. Para perfilar el problema, se identificarán las *fuerzas* que hacen que el problema sea difícil de resolver y restringen el espacio de soluciones posibles.

9.4.1 Contexto

El contexto en el que se desarrolla el patrón consiste en un sistema dinámico compuesto por componentes que interactúan entre sí. La operativa de este sistema consistirá en la realización de una sucesión de tareas por parte de las componentes a lo largo del tiempo, donde la cooperación entre ellas será esencial. Dicho sistema teórico estará dotado con recursos de memoria y procesamiento suficientes.

9.4.2 Problema

El problema que se necesita resolver es cómo evaluar el desempeño real de todo el sistema y sus componentes.

9.4.3 Fuerzas

Las fuerzas identificadas en el contexto anterior son las siguientes:

1 La realidad difiere de la teoría.

Las especificaciones técnicas de un dispositivo no describen completamente su comportamiento. Dos componentes que dispongan del mismo hardware y software instalado y, por tanto, sean teóricamente equivalentes, pueden desembocar en la práctica en diferentes desempeños.

Por ejemplo, si pasamos a hablar de objetos inteligentes o personas, dos entidades inicialmente idénticas pueden diferenciarse en el tiempo como consecuencia de la adaptación a un entorno o por sus mecanismos de aprendizaje. Tales diferencias en el comportamiento sólo son apreciables mediante la observación en el tiempo de su desempeño.

2 Simplificar el catálogo de tareas.

El catálogo de tareas (la descripción de los tipos de tareas que un sistema es capaz de hacer) debería ser definido y mantenido de la forma más concisa y clara posible.

Esto hará que los clientes o usuarios del sistema conozcan sus capacidades de una forma sencilla.

3 Simplificar los diagramas de estados.

La dinámica de una entidad a medida que realiza sus tareas –comportamiento- puede ser descrito como una sucesión de estados. Cuanto más simple sea esta descripción, más fácil será hacer su seguimiento y gestión.

4 El sistema debe ser capaz de detectar a tiempo el mal comportamiento o el rendimiento insuficiente de sus entidades y reaccionar en consecuencia.

5 Se requiere un conocimiento exacto y actualizado de las capacidades (probadas) de las entidades para la asignación de tareas.

6 El sistema debería ser capaz de mostrar evidencias objetivas de su desempeño cuando sea requerido.

7 La solución propuesta debería ser adaptable para que los desarrolladores sean libres de utilizar la aproximación que más se adapte a sus necesidades particulares.

9.4.4 Solución

Los objetivos a conseguir con la solución propuesta son:

- Observar cómo evoluciona el sistema en el tiempo, y deducir su desempeño desde esta observación.
- Definir lo que se necesita medir, identificar los datos que se necesitan reunir para cuantificar el comportamiento del sistema.
- Proporcionar una estructura de persistencia para almacenar esta información.

Para ello, se ha definido el Patrón de Acontecimientos basados en Tareas, representado mediante el diagrama de clases UML de la siguiente figura, que especifica un patrón de diseño

que define una estructura de persistencia para almacenar acontecimientos basados en tareas.

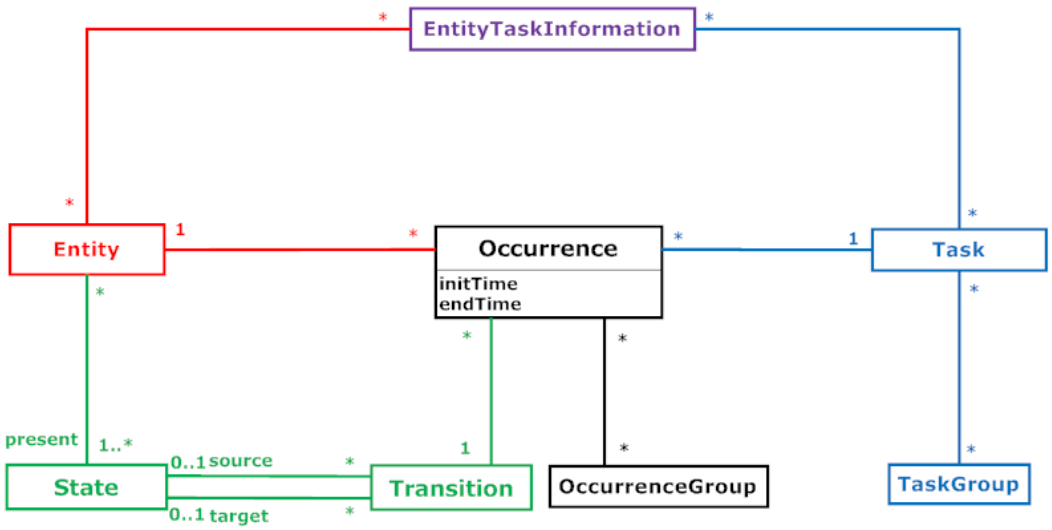


Figura 9-1 Patrón de Acontecimientos basados en Tareas

El principal concepto de este patrón es el de acontecimiento, representado por la clase *Occurrence*. En este caso, a diferencia de los acontecimientos basados en protocolos, se ha optado por representar físicamente la entidad del acontecimiento para incluir en él la información de cuándo ha comenzado y cuándo ha concluido su definición, permitiéndose así la existencia de acontecimientos definidos en un periodo de tiempo.

Por ejemplo, pensemos en una tarea que implique accionar un mecanismo durante un tiempo concreto sin interrupción, como un botón para hacer funcionar un elevador. Incluyendo el concepto de acontecimiento con la información del tiempo, podremos registrar dicho acontecimiento conociendo cuándo se ha empezado a accionar el elevador, y cuándo se ha soltado.

Los elementos con los que está asociado el concepto de acontecimiento corresponden a las tres dimensiones que componen un acontecimiento basado en tareas:

- La clase *Task* (en color azul), representa las tareas ejecutadas –guías- por las entidades cuando ocurren los acontecimientos.
- La clase *Entity* (en color rojo), representa las entidades –componentes, recursos…- del sistema, la estructura.
- La clase *Transition* (en color verde) y la clase *State* (también en verde), representa los cambios de estado producidos por la ejecución de las tareas sobre los posibles estados de las entidades respectivamente, el comportamiento. Las instancias de estas clases pueden ser deducidos de un diagrama de estados UML.

Las tareas pueden ser tareas atómicas que se agrupan en una tarea más compleja, *TaskGroup*. Es el caso por ejemplo del ejemplo de robot de vigilancia comentado anteriormente, donde la tarea de vigilancia es un grupo de tareas: iniciar vigilancia, moverse a un punto, informar de incidencia, finalizar vigilancia…

Respecto a los estados, para una mejor consulta de la actualidad y para una trazabilidad completa de las transiciones (cambios de estado), se incluyen los elementos para conocer los estados actuales de cada entidad y el estado origen y final de las transiciones.

Como extensión a la definición de acontecimiento basado en tareas, se añade la clase *EntityTaskInformation* (en color morado), que incluirá información que relaciona las entidades y las tareas que a veces se necesita. Por ejemplo, en esta relación se pueden incluir las capacidades de una entidad para realizar una determinada tarea (las entidades están dotadas con capacidades, y la ejecución de una tarea requiere de una o varias capacidades).

El patrón también permite considerar grupos de acontecimientos, mediante la clase *OcurrenceGroup*, que permitirá relacionar un conjunto de acontecimientos que han tenido lugar en el tiempo y que cumplen una misma propiedad. Este concepto será de gran ayuda para la extracción de conocimiento relacionado con acontecimientos.

Un claro ejemplo de grupo de acontecimientos es la historia de acontecimientos de una entidad, definida como el conjunto de acontecimientos que han ocurrido relacionados con ella (apartado 3.3).

Otro ejemplo es el conjunto de acontecimientos que tienen lugar durante la ejecución de una tarea. Una tarea, como se puede comprobar en las cardinalidades, puede producir varios acontecimientos.

Los acontecimientos serán almacenados, siguiendo el patrón definido, en una Base de Acontecimientos, la cual será explotada mediante un Sistema Gestor de Bases de Acontecimientos a través de sus herramientas para extraer conocimiento de ellos.

Entre otras cosas, estas herramientas deben permitir la recuperación de información histórica, tal como el conjunto de todos los acontecimientos relacionados con una entidad o el tiempo medio invertido en la realización de una tarea específica.

Por tanto, esta solución permite al sistema buscar a través de la historia de todos sus componentes y extraer conocimiento relacionado con la capacidad de una entidad para realizar una tarea.

9.4.5 Consecuencias

Las consecuencias del patrón se han evaluado identificando los beneficios y desventajas que se consiguen con él, e identificando si se satisfacen las fuerzas definidas en el planteamiento del patrón.

Beneficios

- Permite la monitorización del desempeño real de las entidades, ya que registra lo que realmente realizan las entidades en el tiempo y almacena la información de sus capacidades actuales. Por lo tanto, se cumple la fuerza 1.
- Separa la información relacionada con el Catálogo de Tareas (a través de la clase Tarea) de la relacionada con los estados del sistema y su operación, haciéndolo, por tanto, más claro y conciso (fuerza 2).
- Separa la información relacionada con los estados (a través de la clase State) de la

relacionada con el Catálogo de Tareas y la operación del sistema. Los estados son comunes a todas las tareas, por lo que los diagramas de estados de las entidades se simplifican (fuerza 3).

- Permite la detección temprana de malos comportamientos o desempeños. Por ejemplo, el tiempo que le cuesta a una entidad realizar una tarea puede ser fácilmente obtenible desde la fecha inicial del primer acontecimiento y la fecha final del último acontecimiento que tiene lugar durante la ejecución de dicha tarea. Si dicho periodo excede el límite, el sistema puede reaccionar. Esto está relacionado con la fuerza 4.
- Permite la monitorización de una entidad desde su nacimiento (el momento en el que es introducida en el sistema) hasta el tiempo actual. Esto permite la definición y obtención de indicadores actualizados en cualquier momento, lo que se alinea con la fuerza 5.
- Proporciona a cualquier evaluador (interno o externo) datos a partir de los cuales se pueden calcular indicadores de rendimiento (fuerza 6).
- La solución es metodológicamente y tecnológicamente independiente, por lo que puede ser de aplicación a diferentes contextos (fuerza 7).

Desventajas

- La aplicación del patrón a un sistema requiere la definición precisa de lo que se entiende por 'buen desempeño' en ese contexto específico, y cómo puede ser medido (por ejemplo, se necesitan indicadores de desempeño claramente establecidos).
- Se necesita una buena capacidad de procesamiento y almacenamiento. Los recursos de almacenamiento necesitan incrementarse en el tiempo debido a que en la base de acontecimientos asociada se registrarán todos los acontecimientos que se vayan produciendo.
- Coste: Los costes aumentarán por esta necesidad de potencia de procesamiento adicional, así como por los gastos de gestión adicionales que provoque. Como se ha visto en el apartado 4.6.5 de evaluación de la aplicación del modelo de acontecimientos en el sistema ARASIS, el coste en tiempo de procesamiento crece linealmente con el número de acontecimientos, manteniéndose en unos márgenes eficientes en base a la cantidad de conocimiento gestionada en el sistema.

9.4.6 Pruebas de concepto

En este apartado se realiza la aplicación del patrón para la evaluación del desempeño sobre los tres contextos que se presentaron en la introducción inicial del capítulo, y que se concretan en los siguientes escenarios:

- Empresa que quiere evaluar el desempeño de sus unidades de negocio, como el servicio de atención al cliente desarrollado sobre las buenas prácticas de ITIL [10].
- Sistema de salud que quiere evaluar el rendimiento y la calidad de los resultados de sus médicos que trabajan en Urgencias.
- Sistema de vigilancia con varios robots, donde se quiere evaluar el rendimiento de un robot en la práctica y comprobarlo sobre el presupuesto teórico.

La realización de estas pruebas de concepto nos permite comprobar la adaptabilidad del

patrón definido a diferentes contextos.

9.4.6.1 Gestión de los Servicios de Tecnologías de Información

ITIL es un marco de trabajo ampliamente aceptado que describe unas buenas prácticas para la Gestión de Servicios TI, proporcionando los mecanismos necesarios para lograr la certificación en el estándar ISO 20000 de gestión de servicios TI.

En el Capítulo 5, hemos visto en pequeños ejemplos cómo se puede aplicar el concepto de acontecimiento basado en protocolos a la gestión de servicios TI siguiendo las buenas prácticas de ITIL. En ese capítulo, enfocábamos dicha aplicación desde la necesidad que se tiene en ciertos sistemas de una gestión de incidencias o problemas que pueden surgir en el funcionamiento de los mismos. El objetivo de esta gestión suele ser conocer cómo se han resuelto dichas incidencias y así tener evidencias para medir la calidad del sistema.

En dicho enfoque, los objetos del sistema, y por tanto el núcleo sobre el que se registra todo el conocimiento, son los propios incidentes, problemas, cambios, eventos… y demás elementos definidos en los procesos ITIL. En la base de acontecimientos lo que se registra es la evolución en el tiempo de los mismos, conociendo qué protocolos han actuado sobre ellos, los estados por los que han pasado como consecuencia de dichas ejecuciones… Es decir, el conocimiento almacenado en acontecimientos basados en protocolos. Este enfoque es perfecto para conseguir una gestión de calidad de los servicios TI. Sin embargo, en ciertas situaciones el objetivo que necesitamos conseguir en la gestión de servicios TI es la evaluación del desempeño de las personas o los equipos de personas que realizan y ejecutan dichos procesos de ITIL.

En este caso, si utilizamos la variante que ofrece el enfoque del acontecimiento basado en tareas, mediante el Patrón de Acontecimientos basados en Tareas, podemos obtener un sistema que nos permita esa evaluación del rendimiento de dichos equipos de personas en una organización. Así, en esta aproximación, dichos equipos, o unidades funcionales, serán el núcleo del conocimiento almacenado, y el objetivo será evaluar su desempeño en la ejecución de los procesos ITIL a través del registro de las tareas que han realizado en el sistema.

ITIL cubre todo el ciclo de vida de un servicio, desde su comienzo hasta su finalización, definiendo los diferentes procesos y actividades que se necesitan realizar en todas las etapas de su ciclo de vida. Las etapas que se definen en el ciclo de vida del servicio son:

- Estrategia de Servicio. En esta etapa se establece la visión general y políticas aplicables a los servicios TI de acuerdo con el negocio de la organización. Para ello, en esta etapa se desarrollan procesos como el de Gestión Financiera o el de Gestión del Portfolio de Servicios.
- Diseño de Servicio. El objetivo de esta etapa es diseñar nuevos servicios o modificar los ya existentes en el catálogo de servicios. Ejemplos de procesos en esta etapa son la Gestión del Catálogo de Servicios o la Gestión de Niveles de Servicios.
- Transición del Servicio. Los procesos de esta etapa se encargan de que los servicios definidos en la etapa de diseño lleguen adecuadamente a producción; entre otros procesos, se encuentran la Gestión de Cambios, la Gestión de la Configuración y Activos o la Gestión de Entregas y Despliegues.
- Operación del Servicio. El objetivo de esta etapa es asegurar la correcta prestación de

los servicios, atendiendo solicitudes –proceso de Gestión de Peticiones- resolviendo incidencias y problemas –Gestión de Incidencias y Gestión de Problemas- o controlando el acceso a los servicios –Gestión de Accesos a los Servicios TI-, etc.

- Mejora Continua del Servicio. La función de esta etapa es monitorizar, evaluar y analizar la ejecución de servicios y procesos con el objetivo de elaborar planes de mejoras de los servicios. Para ello se realizan el Proceso de Mejora y el proceso de Informes de Servicios TI.

Cada proceso definido en las etapas anteriores, se compone de un conjunto estructurado de actividades diseñadas para lograr un objetivo específico. Por ejemplo, en el proceso de gestión de incidencias, el objetivo es la resolución de un incidente, y para ello se realizan las actividades de identificación, registro, clasificación, priorización, investigación y diagnóstico, resolución y recuperación, y cierre.

Los encargados de realizar estas actividades son, como se ha comentado anteriormente, las unidades funcionales o funciones en ITIL: un equipo o grupo de personas (y las herramientas que utilizan) especializadas en realizar uno o más procesos y actividades. Por ejemplo, las líneas de atención al cliente del Centro de Servicios –Service Desk-, que se encargan, entre otros procesos, de la gestión de incidencias.

Además, un proceso puede tener relación con otros procesos. Por ejemplo, en la gestión de incidencias, cuando se investiga el incidente y se observa que es recurrente en el tiempo, se puede considerar como un problema y ser escalado a la gestión de problemas para su resolución.

Todos estos conceptos anteriores se han plasmado en el diagrama de clases UML siguiente aplicando el Patrón de Acontecimientos basados en Tareas:

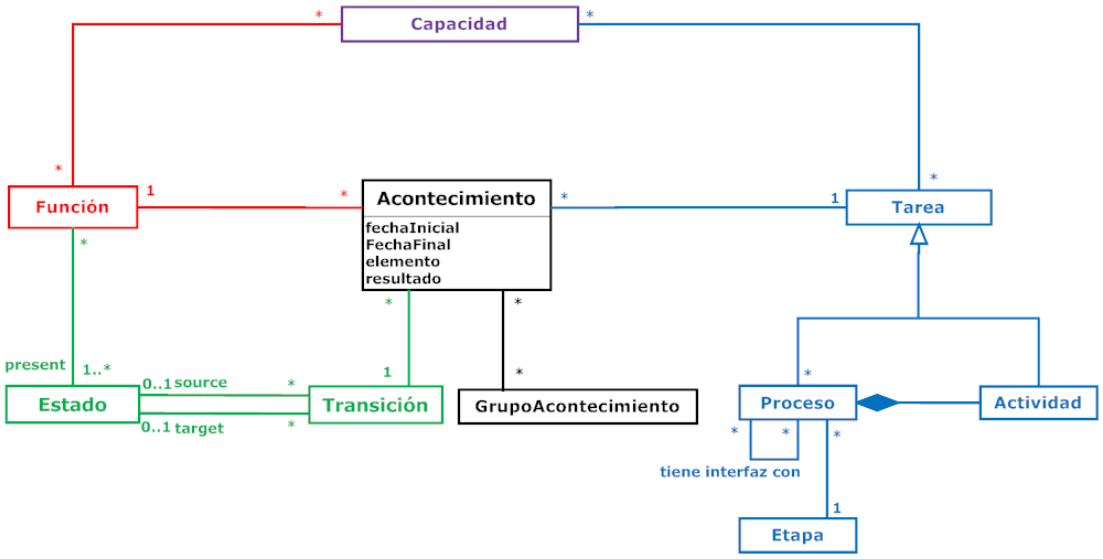


Figura 9-2 Utilización del Patrón de Acontecimientos basados en Tareas en una Gestión de Servicios basada en ITIL

Como se puede ver, la clase entidad del patrón se representa mediante el elemento Función

para almacenar las diferentes unidades funcionales que realizan los procesos ITIL.

Mediante la clase Tarea se modela tanto los procesos como las actividades en las que se desglosan, para permitir distintos grados de granularidad en función de la organización que implemente el patrón. Entre estos dos conceptos se han incluido las distintas relaciones y asociaciones entre procesos y actividades comentadas anteriormente.

Entre la clase Función y la clase Tarea, representando a la clase EntityTaskInformation del patrón, se incluyen las capacidades y habilidades de las funciones para realizar las tareas, mediante el elemento Capacidad.

Cada función dispone de unas capacidades que la cualificarán para realizar una serie de tareas específicas, las cuáles a su vez requerirán de una serie de capacidades para su ejecución. Por ejemplo, según la distribución de funciones que realiza ITIL, la función del Servicio de Asistencia (Service Desk) tiene las capacidades que se requieren para encargarse de los procesos de Gestión de Incidencias y Gestión de Peticiones; por el contrario, la función de Gestión Técnica se encarga del proceso de la Gestión de Problemas cuando la investigación de un fallo afecta la infraestructura TI.

En el siguiente diagrama de estados UML se describen los posibles estados, y transiciones entre ellos, que definen el comportamiento de las funciones:

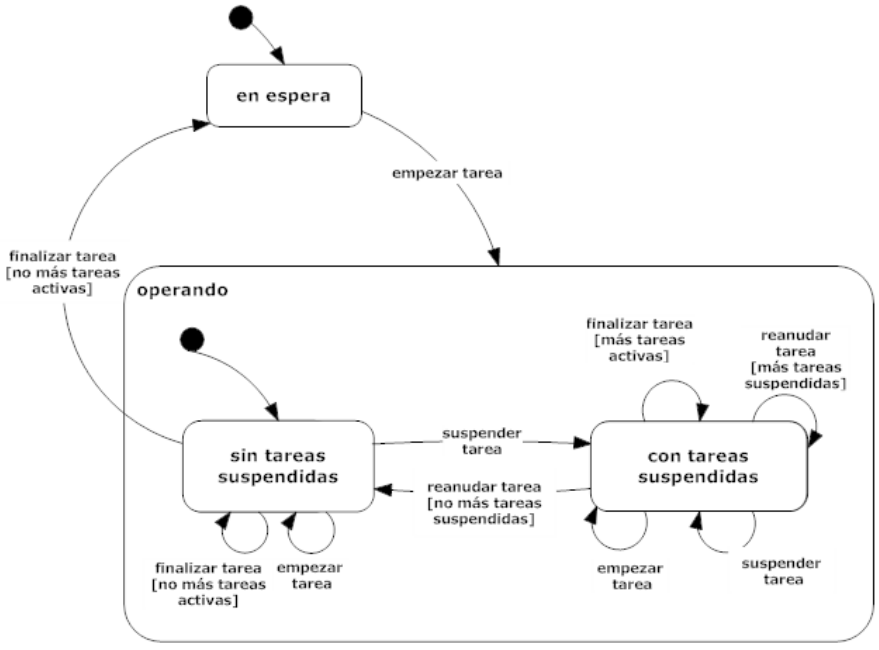


Figura 9-3 Diagrama de estados en una Gestión de Servicios basada en ITIL

Existen dos estados generales, *en espera* y *operando*. En el estado *operando*, una función puede realizar varias tareas simultáneamente. Por ejemplo, cada miembro del equipo de la función de Servicio de Asistencia puede gestionar un incidente diferente al mismo tiempo.

Además, las nuevas tareas pueden empezar en cualquier momento, como resultado tanto de una nueva asignación (por ejemplo, se identifica un nuevo incidente) como de la terminación de una tarea anterior. Un ejemplo de este último caso sería la actividad de clasificación de incidentes, que se realiza una vez se ha realizado la actividad de registro de los mismos.

Por otro lado, en ciertos casos, la realización de una tarea puede ser suspendida. En este caso, la suspensión de la tarea puede ser debida a dos causas distintas:

- La tarea puede ser suspendida por necesidad de la propia función. Por ejemplo, la investigación de un incidente puede escalarse a un grupo de soporte con mayor conocimiento técnico, manteniendo la responsabilidad del incidente la función que lo ha escalado.
- La tarea puede estar suspendida, 'a la espera', cuando la razón por la que no se puede continuar es debida al usuario, no a la función. Por ejemplo, imaginemos un incidente que está resuelto. Para el cierre del incidente se requiere de la conformidad del usuario, pero puede suceder que el Servicio no logre localizarlo y la función de Centro de Servicios suspende la tarea de cierre del incidente hasta que logre localizar al usuario para recabar su conformidad.

El acontecimiento está por tanto relacionado con la transición entre estados mientras una función está realizando una tarea.

Como información a almacenar en el acontecimiento, además de las *fechas* en las que se ha realizado la ejecución de la tarea, se encuentra el *elemento* que se gestiona en la tarea y los *resultados* entregados como resultado de la ejecución de los procesos y actividades. Por ejemplo, el elemento podría ser un incidente concreto que se ha producido y como ejemplo de resultados sería el diagnóstico desarrollado en la actividad de investigación y diagnosis del mismo.

Los acontecimientos que comparten ciertas condiciones se agruparían mediante la clase Grupo de Acontecimientos. Sería el caso de la secuencia de acontecimientos relacionados con un incidente particular, desde su detección hasta su cierre.

Veamos todos estos conceptos y su aplicación a un ejemplo concreto. Siguiendo con el caso de la gestión de incidencias, imaginemos que hay un incidente que está siendo gestionado por el Centro de Servicios; por ejemplo, en una empresa de consultoría informática no está disponible una página web. Llamemos a este incidente *i1*.

Después de dos horas de trabajo de investigación y diagnosis (no tenemos en cuenta las tareas de detección, clasificación…), se ha encontrado una posible solución a las 12:00, reiniciar el servidor. La solución se aplica a las 12:45 y el Centro de Servicios intenta localizar al usuario que ha informado del incidente para su cierre sin éxito en ese momento. El usuario se localiza a las 14:10, dando conformidad a la solución y el Centro de Servicios termina la tarea de cierre del incidente.

En el ejemplo anterior, el Servicio de Asistencia y la Gestión de Incidencias son instancias de Función y Proceso respectivamente, mientras que tenemos tres actividades –clase Actividad-: Investigación y Diagnosis, Resolución y Recuperación, y Cierre del Incidente. Por su parte, el incidente *i1* será el elemento *resultado* de todos los acontecimientos.

Los acontecimientos generados en este ejemplo serán los siguientes:

- Acontecimiento 1, el Centro de Servicios investiga el incidente *i1* (ejecuta la actividad Investigación y Diagnóstico) desde las 10:00 a las 12:00. Como resultado del acontecimiento se obtiene el diagnóstico del incidente. La finalización de la actividad de investigación dispara la transición *finalizar tarea*, quedando la función en el estado *en espera* u *operando* en base a si tienen activas más tareas o no en ejecución.

- Acontecimiento 2. El Centro de Servicios, desde las 12:00 a las 12:45, ejecuta la actividad de Resolución y Recuperación del incidente *i1*. En este caso, el resultado del acontecimiento es que el incidente queda resuelto, y la finalización de la actividad dispara la transición *finalizar tarea*.
- Acontecimiento 3. El Centro de Servicios ejecuta la actividad de Cierre de Incidente del elemento incidente *i1* a las 12:45. Como el usuario no puede ser localizado, se suspende la tarea mediante la transición *suspender tarea*. El Centro de Servicios queda por tanto, si no lo estaba ya, en el estado de *operando*, en el subestado *con tareas suspendidas*.
- Acontecimiento 4. El Centro de Servicios logra contactar con el usuario que ha reportado el incidente. Esto dispara la transición *recuperar tarea* de Cierre de Incidente que estaba suspendida, en el estado *operando*, pasando al subestado *con tareas no suspendidas*.
- Acontecimiento 5. El Centro de Servicios termina de ejecutar la actividad de Cierre del Incidente a las 13:00, disparándose la transición *finalizar tarea* y quedando en el estado *en espera* u *operando* dependiendo de si tiene más tareas en ejecución.

El conocimiento existente en estos acontecimientos nos permitirá definir indicadores de rendimiento como los indicadores TMI - Tiempo Medio entre Incidentes-, TMRS - Tiempo Medio para Restaurar el Servicio y TMF - Tiempo Medio entre Fallos-. Estos indicadores, junto con otros que se pueden definir –ver apartado 9.5- nos permitirán evaluar el desempeño de las funciones en la realización de las tareas.

9.4.6.2 Sistema de salud

En el caso del sistema de salud, vamos a considerar un escenario en el que el objetivo es evaluar el desempeño de los médicos que trabajan en el departamento de urgencias de un hospital. Para ello, vamos a tomar de partida el escenario propuesto en el artículo [26] donde se pretende conocer cómo afecta la multitarea en el rendimiento de los médicos.

En este escenario los médicos van evaluando el estado de los pacientes de forma intercalada, en vez de realizar una atención de forma secuencial.

Por ejemplo, un médico atiende a un paciente y solicita unas pruebas para poder evaluar su estado; mientras espera los resultados, inicia la atención a otro paciente. En este momento, el médico estaría atendiendo a dos pacientes a la vez. Esta forma de procedimiento, habitual en cualquier departamento de urgencias, es lo que se conoce como multitarea en este contexto.

La aplicación del Patrón de Acontecimientos basados en Tareas a este escenario puede verse en la siguiente imagen:

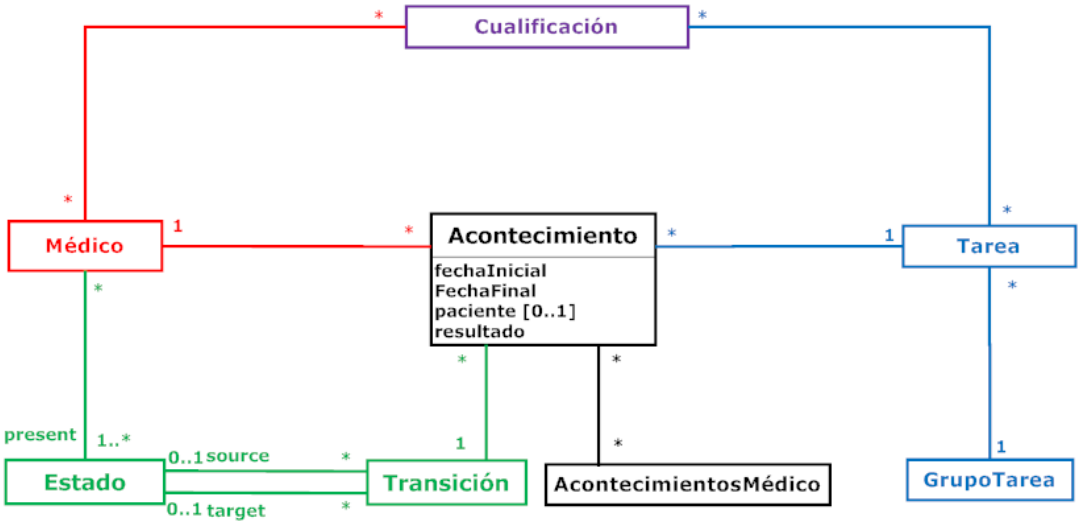


Figura 9-4 Aplicación del Patrón de Acontecimientos basados en Tareas al escenario de [26]

En primer lugar, en este caso las entidades serían los propios médicos, sobre los que queremos medir su desempeño. Cada médico tendrá una serie de conocimientos médicos especializados para desempeñar su trabajo, representadas por la clase 'Cualificación', que le cualifican para desempeñar una tarea, estando autorizado para llevarla a cabo. Las tareas que realizarán los médicos en urgencias a lo largo del tiempo serán, por ejemplo, atender a los pacientes para una exploración inicial, ordenar la realización de pruebas y análisis o prescribir tratamientos terminando la evaluación del paciente. Estas tareas pueden agruparse en grupos de tareas según su propósito, como tareas para el análisis de enfermedades o tareas para su tratamiento. De forma natural, cada tarea por su parte requerirá de una serie de conocimientos médicos específicos para poder ser llevada a cabo, requisitos representados por la asociación de las tareas con la clase Cualificación.

Por otro lado, los estados en los que puede estar un médico, y las transiciones entre ellos, pueden verse en el siguiente diagrama de estados:

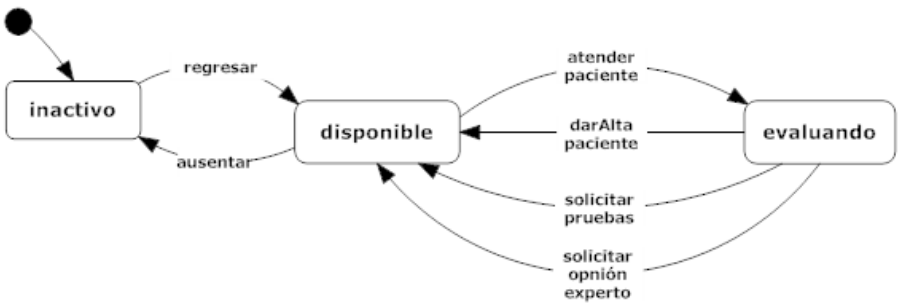


Figura 9-5 Diagrama de estados de un médico en el escenario de [26]

Sobre un médico, es importante saber si está *disponible* para desempeñar su trabajo, o por el contrario está en un estado *inactivo*. Por ejemplo, el médico puede estar de vacaciones, de

baja laboral o ausente por cualquier otro motivo. Además, una vez que está en el hospital, el médico puede estar ocupado evaluando pacientes –estado *evaluando*-. Las transiciones entre el estado *disponible* y el estado *evaluando* surgen por la realización de las tareas, como atender al paciente, prescribir el tratamiento o solicitar pruebas médicas.

La realización de una tarea por parte de un médico, con la consiguiente transición entre los estados del médico, es el acontecimiento basado en tareas, en el cuál se registrará el tiempo que ha transcurrido durante la ejecución de la tarea (a través de la fecha inicial y final), el paciente que ha sido atendido, y el resultado que se obtiene de la ejecución de la tarea. El conjunto de acontecimientos relacionados con la actividad de un médico es lo que se representa con la clase de 'AcontecimientosMédico'.

Veamos todos estos conceptos mediante un ejemplo concreto. Imaginemos la situación con la que hemos explicado el concepto de multitarea, en la que un médico comienza a atender a un paciente que ha entrado a urgencias. El médico atiende al paciente y tras evaluar su estado, solicita la realización de unas pruebas médicas. Cuando recibe los resultados (durante el tiempo de espera ha estado atendiendo a otros pacientes), el médico vuelve con el paciente, evalúa los resultados y prescribe un tratamiento para el paciente dándole el alta.

En este ejemplo, tenemos una instancia de la clase médico y otra de la clase tarea –evaluar paciente-, que se realiza en dos ocasiones. En cuanto a acontecimientos, añadiendo información relativa al tiempo, se registrarían los siguientes (sin contar con los posibles acontecimientos que pudieran surgir del tratamiento de otros pacientes durante la espera de resultados):

- Acontecimiento1. El lunes 17 de octubre de 2016 a las 9:00, el médico M ha quedado disponible y pasa a atender al paciente P. El efecto de este acontecimiento, es que el médico M cambia de estado *disponible* a *evaluando*.
- Acontecimiento2. Desde las 9:00 a las 9:15 el médico M evalúa al paciente P, decidiendo solicitar las pruebas médicas. El efecto de este acontecimiento será que el médico vuelve al estado *disponible* a través de la transición de *solicitar pruebas*. Estas solicitudes serán a su vez el resultado del acontecimiento.
- Acontecimiento3. A las 12:00, una vez que el médico M vuelve a estar disponible de atender a otros pacientes y los resultados de las pruebas ya están disponibles, el médico vuelve a atender al paciente P. El efecto de este acontecimiento es que el estado del médico vuelve a cambiar de *disponible* a *evaluando* al paciente mediante la transición de *atender paciente*.
- Acontecimiento4. Desde las 12:00 a las 12:15, el médico M evalúa los resultados y decide prescribir un tratamiento al paciente P, terminando de diagnosticar al paciente. En este acontecimiento, el médico cambia de estado de nuevo al estado *disponible* a través de la transición dar alta paciente, y el tratamiento prescrito será el resultado.

El conocimiento existente en estos acontecimientos nos permitirá definir indicadores de rendimiento –ver apartado 9.5- que posibiliten la evaluación del desempeño de los médicos, como el indicador TMAP -Tiempo Medio Atención al Paciente- o el número de reevaluaciones de un paciente.

9.4.6.3 Asignación de tareas dinámica en equipos cooperativos de robots

El último escenario en el que vamos a probar la validez del patrón de diseño para la evaluación del desempeño - Patrón de Acontecimientos basados en Tareas-, es el escenario en el necesitamos evaluar el rendimiento de un conjunto de robots que participan en un sistema de vigilancia. El objetivo es que la evaluación del desempeño ayude a optimizar la asignación de tareas. En este contexto, tomaremos como escenario el propuesto en el artículo [85]. En este escenario, se considera un equipo de robots que realizan las funciones de un sistema de vigilancia. Cada robot tiene unas características que lo habilitan para realizar unas tareas específicas, y los robots pueden cooperar de forma simultánea o secuencial.

Para realizar la asignación de las tareas que debe desempeñar cada robot, se debe tener en cuenta tanto sus propias capacidades técnicas como haber realizado correctamente las tareas que le han sido encomendadas. Además, se consideran cuatro situaciones alternativas que pueden darse en la realización de las tareas:

- La tarea T sólo puede ser realizada por el robot R al mismo tiempo.
- Flexibilidad: la tarea T puede ser realizada indistintamente por varios robots.
- Secuencia de tareas y coordinación de robots: la tarea T debe ser realizada primero por el robot R1 y posteriormente por el robot R2.
- Cooperación entre robots: la tarea T debe ser realizada por los robots R1 y R2 simultáneamente.

Teniendo en cuenta las características anteriores del escenario, la asignación de tareas puede reducirse a un problema de optimización, donde la función a optimizar, la función de utilidad, cuantificará la capacidad de un robot para realizar una determinada tarea, y resultará de la compensación entre las cualidades del robot y los costes que implicaría que la tarea la hiciera dicho robot.

Una vez planteado el escenario, la aplicación del patrón a él resultaría en el siguiente diagrama de clases UML:

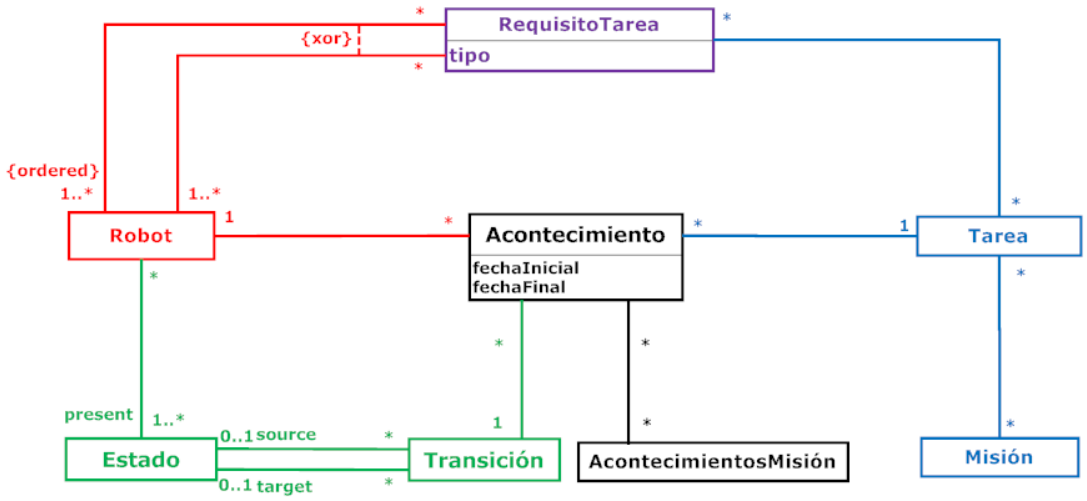


Figura 9-6 Aplicación del Patrón de Acontecimientos basados en Tareas al escenario de [85]

Por un lado, el elemento entidad, sobre el cual queremos evaluar su desempeño, es cada robot del sistema de vigilancia. Por el otro lado, estarán las tareas que realizarán los robots, como 'Comenzar vigilancia', 'Terminar vigilancia', 'Avisar de fallo en la vigilancia', 'Reanudar vigilancia'… Estas tareas estarán agrupadas en misiones que deben realizarse como parte del sistema de vigilancia. Cualquier información relacionada con los requisitos para completar una tarea, como restricciones de tipo de robots y orden (secuencial o concurrente) está modelizada por la clase RequisitoTarea, junto con su atributo *tipo* y las restricciones en las asociaciones con la entidad robot.

Los estados por los que pasarán los robots mientras ejecutan las tareas se describen mediante el siguiente diagrama de estados, definido en el propio escenario tomado como referencia:

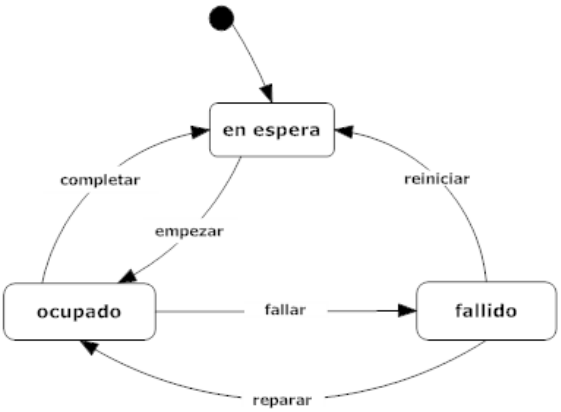


Figura 9-7 Diagrama de estados de un robot en el escenario de [85]

Un robot R realizando una tarea T podrá estar en tres estados diferentes: *en espera*, *ocupado* o *fallido*. Las transiciones entre los estados vendrán definidas por las acciones que pueden surgir al realizar una tarea: *empezar*, *completar*, *fallar*, *reparar* y *reiniciar*.

Por ejemplo, inicialmente un robot estará en el estado *en espera*. Cuando comienza a realizar una tarea, se disparará la transición *empezar*, cambiando al estado *ocupado*. En este estado podrán ocurrir dos situaciones:

- Que complete la tarea quedando de nuevo en espera para nuevas tareas (transición *completar* al estado *en espera*).
- Que falle en la ejecución de la tarea, quedando en el estado *en fallo* mediante la transición *fallar*.

Si el robot está en el estado *en fallo*, podrá ocurrir que se repare retomando la realización de la tarea (transición *reparar* al estado *ocupado*), o que sea reiniciado para comenzar nuevas tareas (transición *reiniciar* al estado *en espera*).

La ejecución de las tareas por parte de los robots, por tanto, hará que éstos cambien el estado en el cual se encuentran en el sistema, con lo que obtenemos los tres elementos que definen el acontecimiento basado en tareas: robot, tarea y efecto. El conjunto de acontecimientos registrados durante la ejecución de una misión es lo que representa la clase 'AcontecimientosMisión'.

Veámoslo con un ejemplo. Imaginemos una situación en la que un robot comienza a vigilar una zona. En un momento determinado, avisa de que se ha atascado en un obstáculo, fallando en la vigilancia. Tras la reparación por los técnicos, reanuda la vigilancia de la zona y completa su misión al final del turno de vigilancia. En este ejemplo, tendríamos los siguientes acontecimientos:

- Acontecimiento 1. El robot R, que lleva una hora en espera, comienza una tarea de vigilancia a las 10:15 del lunes 08 de noviembre. El efecto de este acontecimiento será que el robot pasa al estado *ocupado* a través de la transición *empezar*.
- Acontecimiento 2. A las 11, el robot R avisa de un fallo en la vigilancia. En este acontecimiento, el efecto es que el robot pasará al estado *en fallo* mediante la transición *fallar*.
- Acontecimiento 3. A las 11:15 el robot R reanuda la vigilancia, pasando al estado *ocupado* de nuevo mediante la transición *reparar*.
- Acontecimiento 4. A las 12:00, el robot R termina la tarea de vigilancia. En este acontecimiento, el robot cambiará al estado *en espera* a través de la transición *completar*.

La línea de vida de un robot serán todos los acontecimientos que ha generado el robot en cuestión. En el ejemplo, los cuatro acontecimientos anteriores formarán parte de la historia del robot R.

Si recordamos el objetivo del escenario, era conseguir una función de utilidad que nos permita evaluar el rendimiento de los robots en la realización de las tareas para realizar una asignación más óptima (función $ability_{jk}$ en el artículo). Mediante el uso del patrón, la función de utilidad de un robot R realizando una tarea T puede ser inferida, por ejemplo, con un indicador de rendimiento –ver apartado 9.5- que mida la historia de éxitos y fracasos que ha obtenido en la ejecución de dicha tarea.

9.5 Evaluación del rendimiento con el patrón: indicadores de desempeño

Los acontecimientos almacenados siguiendo el patrón anterior permiten extraer información relacionada con la capacidad de una entidad para realizar una tarea. Por ejemplo, el tiempo medio utilizado por una entidad en la realización de una tarea específica.

En particular, el patrón permite definir dos funciones generales sobre los acontecimientos registrados en la historia de una entidad:

1 Función que calcula el **número de acontecimientos** en los que cualquier entidad pasa de un estado a otro debido a una transición dada. Opcionalmente, se puede restringir la búsqueda a aquellos acontecimientos relacionados con una entidad, una tarea o un grupo de tareas determinado. Especificaremos esta función mediante el símbolo **#**.

2 Función que calcula el **tiempo total acumulado** que cualquier entidad ha estado en un estado, computando la suma de todos los intervalos de tiempo pasados en dicho estado. Cada intervalo se determina por dos acontecimientos: las transiciones que llevan o sacan a la entidad de ese estado. Al igual que en la función anterior, se puede

restringir la búsqueda, opcionalmente, a una entidad, una tarea o un grupo de tareas determinado. Especificaremos esta función mediante el símbolo **Δ**.

La especificación formal de estas dos funciones, utilizando la notación formal BNF (Backus–Naur Form [49])[1] , sería la siguiente:

1 #([<entity>,] < transition > [,< taskSpecifierGroup >])

2 Δ([<entity>,] < state > [,< taskSpecifierGroup >])

donde

<taskSpecifierGroup> ::= <taskSpecifier> {, <taskSpecifier>}

<taskSpecifier> ::= <task> [, <taskGroup>] | <taskGroup>

Las funciones anteriores permiten definir indicadores de rendimiento sobre la historia de las entidades. Los indicadores podrán ser de dos tipos:

- Indicadores de estado. Son aquellos indicadores en los que se necesita especificar, como parámetro del indicador, un estado de la entidad para su cálculo. Opcionalmente, en estos indicadores también se puede establecer una entidad, una tarea o un grupo de tareas para restringir el cálculo del indicador. Un ejemplo de este tipo de indicadores sería un indicador que calcule la fracción de tiempo que una entidad pasa en un determinado estado.
- Indicadores genéricos. Son aquellos indicadores en los que no se requiere la especificación de un estado de la entidad para su cálculo. Opcionalmente, al igual que en el caso anterior, se puede establecer una entidad, una tarea o un grupo de tareas para restringir el cálculo del indicador. Ejemplos de este tipo de indicadores podrían ser el cálculo del número de asignaciones de una entidad a una tarea o el tiempo medio que a una entidad le cuesta completar una tarea.

La especificación formal de los indicadores, utilizando notación BNF, sería mediante funciones con la siguiente sintaxis:

<indicator>::= <stateIndicator> | <genericIndicator>

donde

<stateIndicator> ::= <sIType> (<stateSpecifier>)

<genericIndicator > ::= <gIType> ([<genericSpecifier >])

<stateSpecifier> ::= <state> [,<entity>] [,<taskSpecifierGroup>]

<genericSpecifier>::= <entity> | <taskSpecifierGroup> | <entity>, <taskSpecifierGroup>

Los elementos *gIType* y *sIType* representan el nombre de los diferentes indicadores genéricos y de estado, respectivamente, que serán definidos según el contexto en el que se aplique el patrón. En los siguientes subapartados se muestran ejemplos de indicadores de rendimiento definidos en los escenarios utilizados para las pruebas de concepto del apartado 9.4.6. Algunos de los ejemplos que presentaremos son en esencia los mismos indicares adaptados al contexto en cuestión, mientras que otros son específicos de cada caso.

[1] El significado de los símbolos de la notación BNF que se ha utilizado es el siguiente: definición (::=), nombre de categoría (<>), función lógica 'o' (|), opcionalidad ([ ]) y repetición de cero o más elementos ({ }).

9.5.1 Indicadores en el escenario de Gestión de los Servicios de Tecnologías de Información

El objetivo de este escenario era evaluar el desempeño de las funciones en la ejecución de las tareas de los procesos ITIL. A partir de los acontecimientos almacenados según el patrón de diseño, se pueden definir indicadores que permitan medir el rendimiento de las funciones.

A modo de ejemplo, se podrían definir los siguientes indicadores:

- nAsignaciones. Número de veces que una tarea ha sido asignada a una función.

  Un ejemplo muy característico de aplicación de este indicador puede ser calcular el número de incidentes que ha gestionado una función, al que se le podría dar un nombre propio: nIncidentes. Este valor se obtendría calculando el número de asignaciones del proceso *ProcesoGestiónIncidentes* que ha tenido la función.
- nInterrupciones. Número de veces que una función ha tenido que suspender una tarea específica. Este indicador también se puede generalizar para medir el número de veces que una función ha suspendido cualquier tarea que haya podido realizar.
- tasaÉxito. Tasa de resolución con éxito de una tarea para una función específica. De la misma forma que en el indicador de asignaciones, se puede definir la aplicación de este indicador al caso de los incidentes, llamándole, por ejemplo, tasaÉxitoIncidentes. En esta aplicación, el objetivo sería medir específicamente la tasa de éxito de una función en la resolución de incidentes.
- TMT, Tiempo Medio entre Tareas. Tiempo medio entre dos ejecuciones consecutivas de una tarea.

  Un ejemplo de aplicación de este indicador es el indicador definido en ITIL como Tiempo Medio entre Incidentes –TMI- (MTBI, *Mean Time Between Incidents*), que mide el tiempo medio entre incidentes del servicio. El valor de esta aplicación del indicador se podrá obtener calculando el TMT para la tarea que representa el proceso *ProcesoGestiónIncidentes*.
- TMRS, Tiempo Medio para Restaurar el Servicio (MTRS, *Mean Time to Restore Service*). Tiempo medio que se requiere para recuperar el servicio después de un incidente.
- TMF, Tiempo Medio entre Fallos (MTBF, *Mean Time Between Failures*). Tiempo medio que el servicio está operando sin que se produzca un fallo. El valor de este indicador puede obtenerse mediante los dos indicadores anteriores, utilizando la aplicación del indicador TMT respecto a los incidentes –TMI-, mediante la fórmula: TMF= TMI-TMRS. Este indicador nos proporcionará una medida de la fiabilidad del funcionamiento del sistema; cuanto mayor sea su valor, más fiable será el sistema (más tiempo transcurrirá entre incidentes en el sistema).
- TMCT, Tiempo Medio para Completar una Tarea (MTTC, *Mean Time to Task Completion*). Tiempo medio que le cuesta a una función completar una tarea. Este indicador es una medida de la eficiencia de una función.
- fracciónTiempo: Fracción de tiempo que una función ha pasado en un estado concreto.

En la siguiente tabla se indica el cálculo de los indicadores anteriores utilizando las funciones generales definidas previamente: número de acontecimientos # y de tiempo total en un estado Δ.

Indicador	Formulación
nAsignaciones (función, tarea)	$\#(\text{función}, \textit{empezar tarea}, \text{tarea})$
nAsignaciones (función, actividad, proceso)	$\#(\text{función}, \textit{empezar tarea}, \text{actividad}, \text{proceso})$
nIncidentes (función)	$\text{nAsignaciones}(\text{función}, \textit{procGestIncidentes}) = \#(\text{función}, \textit{empezar tarea}, \textit{procGestIncidentes})$
nInterrupciones (función)	$\#(\text{función}, \textit{suspender tarea})$
nInterrupciones (función, tarea)	$\#(\text{función}, \textit{suspender tarea}, \text{tarea})$
tasaÉxito (función)	$\frac{\#(\text{función}, \textit{finalizar tarea})}{\#(\text{función}, \textit{empezar tarea})}$
tasaÉxito (función, tarea)	$\frac{\#(\text{función}, \textit{finalizar tarea}, \text{tarea})}{\#(\text{función}, \textit{empezar tarea}, \text{tarea})}$
tasaÉxitoIncidentes (función)	$\text{tasaÉxito}\ (\text{función}, \textit{procGestIncidentes}) = \frac{\#(\text{función}, \textit{finalizar tarea}, \textit{procGestIncidentes})}{\#(\text{función}, \textit{empezar tarea}, \textit{procGestIncidentes})}$
TMT (tarea)	$\frac{\sum_{\text{estado}\ \in\{\textit{operando}\ ,\textit{en espera}\ \}} \Delta(\text{estado}, \text{tarea})}{\#(\ \textit{empezar tarea}, \text{tarea})}$
TMI ()	$\text{TMT}(\textit{procGestIncidentes}) = \frac{\sum_{\text{estado}\ \in\{\textit{operando}\ ,\textit{en espera}\ \}} \Delta(\text{estado}, \textit{procGestIncidentes})}{\#(\ \textit{empezar tarea}, \textit{procGestIncidentes})}$
TMRS ()	$\frac{\Delta(\textit{operando}, \textit{procGestIncidentes})}{\#(\ \textit{empezar tarea}, \textit{procGestIncidentes})}$
TMF ()	TMI - TMRS
TMCT (función, tarea)	$\frac{\Delta(\text{función}, \textit{operando}, \text{tarea})}{\#(\ \text{función}, \textit{empezar tarea}, \text{tarea})}$
fracciónTiempo (función, estado)	$\frac{\Delta(\text{función}, \text{estado})}{\sum_{\text{estado}\ \in\{\textit{operando}\ ,\textit{en espera}\ \}} \Delta(\text{función}, \text{estado})}$
fracciónTiempo (función, estado, tarea)	$\frac{\Delta(\text{función}, \text{estado}, \text{tarea})}{\sum_{\text{estado}\ \in\{\textit{operando}\ ,\textit{en espera}\ \}} \Delta(\text{función}, \text{estado}, \text{tarea})}$

Figura 9-8 Indicadores de rendimiento para el escenario de ITIL

En las fórmulas anteriores, se han escrito en cursiva las transiciones y estados concretos del diagrama de estados del escenario. En los indicadores que se calculan para el caso particular de los incidentes la función generalmente siempre será el Servicio de Atención –*Service Desk* en inglés-, sin embargo, se incluye la función como parámetro para que la especificación de los indicadores sea más general.

Al igual que se ha hecho con los incidentes, también se pueden definir otras medidas de rendimiento para los procesos y actividades más importantes de la gestión de los servicios TI, como cuánto tiempo se tarda de media en aplicar cambios en el sistema, cuánto tiempo se tarda en aprobar dichos cambios de media, etc.

9.5.2 Indicadores en el escenario del sistema de salud

En el caso del sistema de salud, el objetivo era definir los indicadores que nos van a permitir evaluar el desempeño de los médicos en una evaluación multitarea de los pacientes, con el conocimiento que nos proporciona la información almacenada utilizando el patrón.

Por ejemplo, podríamos definir los siguientes indicadores:

- nAtenciones. Número de veces que un médico atiende a un paciente. Este indicador calcula el total de atenciones del médico, es decir, sin importar si el médico atiende al mismo paciente varias veces.
- nAusencias. Número de veces que un médico se ha ausentado.
- fracciónTiempo: Fracción de tiempo que un médico ha pasado en un estado concreto. Este indicador nos permitiría obtener, por ejemplo, el tiempo que un médico ha estado disponible para la atención de pacientes pero sin efectuar dicha tarea, dándonos una medida de la carga de trabajo del médico.
- TMAP, Tiempo Medio de Atención a Pacientes. Tiempo medio que dedica un médico a la atención de pacientes. Este indicador permitiría analizar si un médico dedica un tiempo adecuado de media a atender los pacientes.
- tasaProductividadPorAtenciones. Ratio de productividad de un médico, entendido como el número de pacientes dados de alta por un médico respecto al número de atenciones realizadas.
- tasaSolicitudPruebas. Indicador que mide la cantidad de pruebas que un médico solicita respecto al número de atenciones que realiza.

A continuación se muestra la formulación de estos indicadores utilizando las funciones de número de acontecimientos # y tiempo total en un estado Δ.

Indicador	Formulación
nAtenciones (médico)	$\#(\text{médico}, atender\ paciente)$
nAusencias (médico)	$\#(\text{médico}, ausentar)$
fracciónTiempo (médico, estado)	$\frac{\Delta(\text{médico}, \text{estado})}{\sum_{\text{estado}\ \in\{inactivo\ ,disponible\ ,evaluando\ \}} \Delta(\text{médico}, \text{estado})}$
TMAP (médico)	$\frac{\Delta(\text{médico}, evaluando)}{\#(\ \text{médico}, atender\ paciente)}$
tasaProductividad (médico)	$\frac{\#(\text{médico}, darAlta\ paciente)}{\#(\text{médico}, atender\ paciente)}$
tasaSolicitudPruebas (médico)	$\frac{\#(\text{médico}, solicitar\ pruebas)}{\#(\text{médico}, atender\ paciente)}$

Figura 9-9 Indicadores de rendimiento para el escenario de salud

En las fórmulas anteriores, se han escrito en cursiva las transiciones y estados concretos del diagrama de estados del escenario.

La obtención de uno o varios de estos indicadores nos permitirá realizar la evaluación del

desempeño de un médico. Por ejemplo, podríamos conocer que el tiempo medio de atención completa de un médico es muy alto comparado con otro, pero que su tasa de productividad es también más alta (lo que indicaría más eficiencia en el diagnóstico; una tasa de productividad muy baja puede deberse a que el médico requiera reevaluar constantemente a sus pacientes). O también, podríamos conocer, por ejemplo, que un médico realiza muchas pruebas a sus pacientes, pero que por el contrario el tiempo medio de evaluación al paciente es muy bajo, lo que implica que quizás podría reducir el número de solicitudes de pruebas (y por tanto del consiguiente coste) si aumentara ese tiempo de evaluación.

9.5.3 Indicadores en el escenario de asignación de tareas dinámica en equipos cooperativos de robots

Si recordamos el objetivo del escenario de robots, éste era conseguir una función de utilidad que nos permita evaluar el rendimiento de los robots en la realización de las tareas para realizar una asignación más óptima (función $ability_{jk}$ en el artículo [85]).

Para ello, se pueden definir los siguientes indicadores:

- nAsignaciones, número de asignaciones. Cantidad de veces que una tarea se ha asignado a un robot.

  Para obtener este valor bastaría contar el número de acontecimientos existentes que involucren al robot y la tarea indicados.
- nFallos, número de fallos de un robot. Cantidad de fallos que un robot ha cometido en la realización de tareas.

  El indicador será resultado de contar el número de acontecimientos existentes que involucren al robot en transiciones *fallar*.
- tasaÉxito. Cantidad de veces que un robot termina de forma exitosa una tarea.

  Este valor se obtendrá relacionando, para un robot y tarea dados, el número de acontecimientos existentes en una transición de *completar* respecto a los acontecimientos existentes en transiciones *comenzar.*
- TMF, Tiempo Medio entre Fallos (MTBF, *Mean Time Between Failures*). Tiempo medio en el que el robot está efectivamente realizando la tarea sin fallo.
- TMR, Tiempo Medio para ser Reparado (MTTR, *Mean Time To Repair*). El tiempo medio que se requiere para restaurar un robot a su modo de operación normal después de un fallo. Este indicador es una medida de la mantenibilidad del robot.
- TMCT, Tiempo Medio para Completar Tarea (MTTC, *Mean Time to Task Completion*). Tiempo medio que un robot necesita para completar una tarea.
- fracciónTiempo: Tiempo pasado por un robot en un estado, calculable computando el tiempo existente en los acontecimientos del estado a medir.

En la siguiente tabla se muestran los indicadores anteriores utilizando las funciones de número de acontecimientos # y tiempo total en un estado Δ:

Indicador	Formulación
nAsignaciones (robot, tarea)	$\#(\text{robot}, \textit{empezar}, \text{tarea})$
nAsignaciones (robot, tarea, misión)	$\#(\text{robot}, \textit{empezar}, \text{tarea}, \text{misión})$
nFallos (robot)	$\#(\text{robot}, \textit{fallar})$
tasaÉxito (robot)	$\frac{\#(\text{robot}, \textit{completar})}{\#(\text{robot}, \textit{empezar})}$
tasaÉxito (robot, tarea, misión)	$\frac{\#(\text{robot}, \textit{completar}, \text{tarea}, \text{misión})}{\#(\text{robot}, \textit{empezar}, \text{tarea}, \text{misión})}$
TMF (robot)	$\frac{\Delta(\text{robot}, \textit{ocupado})}{\#(\text{robot}, \textit{fallar})}$
TMF (robot, tarea, misión)	$\frac{\Delta(\text{robot}, \textit{ocupado}, \text{tarea}, \text{misión})}{\#(\text{robot}, \textit{fallar}, \text{tarea}, \text{misión})}$
TMR (robot)	$\frac{\Delta(\text{robot}, \textit{fallido})}{\#(\text{robot}, \textit{fallar})}$
TMCT (robot, tarea)	$\frac{\Delta(\text{robot}, \textit{ocupado}, \text{tarea})}{\#(\text{ robot}, \textit{empezar}, \text{tarea})}$
fracciónTiempo (robot, estado)	$\frac{\Delta(\text{robot}, \text{estado})}{\sum_{\text{estado } \in \{\textit{ocupado}, \textit{en espera}, \textit{fallido}\}} \Delta(\text{robot}, \text{estado})}$
fracciónTiempo (robot, estado, tarea, misión)	$\frac{\Delta(\text{robot}, \text{estado}, \text{tarea}, \text{misión})}{\sum_{\text{estado } \in \{\textit{ocupado}, \textit{en espera}, \textit{fallido}\}} \Delta(\text{robot}, \text{estado}, \text{tarea}, \text{misión})}$

Figura 9-10 Indicadores de rendimiento para el escenario de robots

Como en los escenarios anteriores, en las fórmulas se han escrito en cursiva las transiciones y estados concretos del diagrama de estados del escenario.

La función de utilidad del artículo se puede inferir a partir de los indicadores definidos mediante los acontecimientos almacenados usando el patrón. Por ejemplo, se podría inferir del indicador tasaÉxito, optimizando la asignación de tareas en función del menor número de fallos que los robots incurren en la realización de las tareas a asignar.

9.5.4 Utilización de los indicadores en la asignación de tareas

Los indicadores de rendimiento definidos en este apartado permiten realizar mediciones cuantificables de la historia del desempeño de las entidades en la ejecución de las tareas, obteniendo sus resultados mediante el conocimiento almacenado en los acontecimientos registrados mediante el patrón definido en este capítulo.

Tener información sobre el desempeño en el pasado permite mejorar las futuras asignaciones de tareas, tal y como demuestran, por ejemplo, los métodos desarrollados en [81][39]. En estas propuestas, se desarrollan mecanismos de *data mining* para obtener información del

pasado y mejorar la asignación de tareas utilizando diferentes técnicas, como árboles de decisión. En nuestro caso, este conocimiento se obtiene utilizando para ello un modelo, los acontecimientos, que se puede explotar de forma eficiente, tal y como se ha visto en el Capítulo 7 en la evaluación de la implementación del caso ARASIS. Este modelo, además, almacena toda la información necesaria para conocer qué ha ocurrido en la ejecución de las tareas.

Por tanto, mediante los indicadores de rendimiento, definidos sobre los acontecimientos basados en tareas, tenemos un mecanismo de ajuste de la optimización de las asignaciones utilizando la historia de desempeño de los recursos. La asignación de tareas utilizando los indicadores se puede realizar en dos vertientes, que pueden ser complementarias entre sí según sea la situación y las necesidades:

- Utilizar exclusivamente la información de los indicadores de rendimiento, realizando la asignación en función de límites establecidos con los indicadores.

  La utilización exclusiva de los indicadores de rendimiento es útil en situaciones en las que no se tiene, o no se puede tener de forma sencilla, información acerca del coste, o un valor a optimizar, que puede suponer que un recurso realice una determinada tarea, o este coste es igual entre todos los recursos impidiendo una distinción entre ellos.

  Un ejemplo de esta aplicación, lo hemos realizado en el apartado 9.2, donde la asignación de la resolución de una incidencia se realizaba por un indicador de fiabilidad en la resolución.

  Otro ejemplo similar lo podríamos tener en el escenario anterior del equipo de vigilancia multirobots, en el caso de que todos los robots fueran iguales en capacidades técnicas, el elemento diferenciador entre unos y otros, para la función de utilidad, serían los indicadores de rendimiento que se pueden extraer de la información de los acontecimientos almacenados, como la tasa de éxito/errores.

- Ajustar los métodos de optimización habituales, optimizando los valores de entrada a los métodos de resolución, como los costes, con los indicadores de rendimiento.

  Teóricamente, con la información de los indicadores se puede mejorar y precisar los datos de entrada a los métodos de optimización habituales, como los métodos de programación lineal, en los que el método se base en optimizar la asignación utilizando una variable o matriz de costes de información fija del sistema. Por ejemplo, imaginemos que se utiliza como información el coste/hora de los recursos. Si en vez de utilizar sólo este valor, se utilizan los indicadores del tipo del indicador TMCT –Tiempo Medio para Completar una Tarea-, podemos precisar los datos de entrada a los métodos de optimización obteniendo el coste real probable que implicará que cada recurso haga una determinada tarea, multiplicando el valor de coste/hora por la media de horas que invierte el recurso en la tarea. De este modo se tiene un dato más preciso de cara a minimizar el coste real en la ejecución de las tareas. Una vez optimizados los valores de entrada, los algoritmos se ejecutan según su método de optimización, por lo que se aprovecha toda la potencia de optimización que proporcionan.

Como línea de trabajo futura se encuentra la aplicación de nuestro modelo en entornos de trabajo reales, haciendo uso de diferentes métodos de asignación, con objeto de analizar qué mejoras se obtienen en la asignación de tareas.

V. CONCLUSIONES Y TRABAJO FUTURO

Capítulo 10. Conclusiones y trabajo futuro

Los objetivos que se plantearon para la realización de esta tesis se pueden resumir en tres grandes puntos:

- Desarrollar una base teórica que permita evolucionar la consideración clásica de las bases de datos, definiendo conceptos, modelos y estructuras de persistencia que posibiliten almacenar, como estructura atómica mínima de información, todos los datos necesarios sobre un hecho que acontezca en un sistema de información: qué ha ocurrido, quién lo ha realizado y cuándo se ha producido.
- Desarrollar una estrategia de diseño, utilizando la base teórica anterior, para construir sistemas que permitan monitorizar y analizar de una forma eficiente la evolución de la información que se genera en la ejecución de los procesos de negocio de una organización.
- Desarrollar un patrón que permita evaluar el desempeño de los recursos de la empresa que ejecutan los procesos de negocio, proporcionando el conocimiento para mejorar la asignación de las tareas de los procesos a los recursos en función de la historia de su desempeño.

Estos objetivos nos permitían abordar la optimización de los procesos de negocio en base a dos enfoques: (1) Optimizar el propio proceso de negocio proporcionando un sistema de monitorización eficiente que facilite el análisis de su ejecución para su optimización, y (2) optimizar la ejecución de las tareas que componen el proceso mejorando su asignación a los recursos adecuados.

A continuación, se muestran las conclusiones en cada uno de los tres grandes bloques en los que se divide la tesis.

Conceptos de acontecimiento, base de acontecimientos y sistema gestor de bases de acontecimientos

El concepto de acontecimiento definido en esta tesis permite reunir información sobre cualquier acción que suceda en un sistema de información: qué ha ocurrido, cuándo y por quién. Para ello, un acontecimiento se define como una estructura tridimensional en base a tres dimensiones: guía –aconteceres-, estructura –objetos- y comportamiento –estados, cambios de estado, datos longitudinales-, de modo que esta información se almacena como unidad mínima en los sistemas de información desarrollados utilizando el concepto de acontecimiento.

El concepto de acontecimiento desarrollado en esta tesis, abarca aquellos acontecimientos artificiales que suceden en entornos gestionados por sistemas de información. Estos acontecimientos son aquellos en los que se puede identificar el elemento –el objeto- de la

acción de forma concreta, y la acción se puede estructurar en una serie de pasos o actividades.

Además, este concepto de acontecimiento se ha definido de forma genérica, como reflejo conceptual de lo sucedido en un sistema, siendo aplicable a los dos enfoques de optimización de los procesos de negocio abarcados en esta tesis, mediante su concreción como acontecimiento basado en protocolos o acontecimiento basado en tareas.

Por otro lado, el concepto de acontecimiento es representable mediante una tupla de tres elementos (a, o, e) donde *a* es la representación de la acción, el acontecer, que actúa sobre o es realizada por un objeto *o,* provocándole el efecto *e*.

Como estructura de persistencia de los acontecimientos, se ha definido el concepto de *base de acontecimientos*, como 'un conjunto estructurado de acontecimientos', donde el acontecimiento es la unidad mínima de información.

Se establece como norma general de una base de acontecimientos, que los acontecimientos registrados en el tiempo no serán eliminados, de forma que se preserve en la medida de lo posible el *provenance* de los datos. Sin embargo, se permitirá la eliminación de acontecimientos para dar cabida a ciertas situaciones que pueden suceder en los sistemas de información, como requerimientos legales de eliminación de información almacenada.

Para gestionar y explotar la información de una base de acontecimientos se define de forma genérica el concepto de *sistema gestor de bases de acontecimientos* –SGBA-, como 'sistema que permita la gestión completa de acontecimientos tanto en términos de su almacenamiento (mediante el uso de una Base de Acontecimientos) como en términos de las herramientas que faciliten la explotación de su conocimiento relacionado'.

Se establece que un SGBA, para ser considerado como tal, debe proporcionar, además de la propia gestión de consulta y registro de los acontecimientos, los mecanismos necesarios para analizar y explotar el conocimiento almacenado en los acontecimientos, permitiendo establecer y obtener las relaciones que existan entre ellos. Entre estos mecanismos, se determina que todo SGBA debe poder proporcionar, como mínimo, la *línea de vida* y la *historia* de un objeto.

Debido a esta capacidad de explotación y gestión de la información de los objetos a través de sus acontecimientos, un SGBA es un sistema longitudinal, ya que es capaz de informar lo que le ha ocurrido a un objeto en un momento puntual del tiempo o durante un periodo.

Por último, como conclusión de todo lo anterior, en los conceptos de acontecimiento, base de acontecimientos y sistema gestor de bases de acontecimientos, tenemos la evolución de la consideración clásica de las bases de datos:

- El acontecimiento como unidad mínima de información, a semejanza del dato.
- La base de acontecimientos como su estructura de persistencia, a semejanza de la base de datos.
- El sistema gestor de bases de acontecimientos –SGBA- como el sistema para gestionar y explotar la información –los acontecimientos- de la base de acontecimientos, a semejanza de los sistemas gestores de bases de datos –SGBD-.

Estrategia de diseño de sistemas de monitorización basados en acontecimientos

Los sistemas de monitorización, llamados BAM -Business Activity Monitoring- habitualmente registran la información mediante sistemas que impiden una eficiente explotación de dicha información (ficheros de texto plano –logs- o estructuras lineales de bases de datos) o no recogen toda la información necesaria para una correcta monitorización y análisis.

La estrategia de diseño que hemos presentado nos permite construir sistemas de monitorización utilizando la teoría de acontecimientos, como sistemas gestores de bases de acontecimientos, que solucionan este tipo de problemas en los sistemas BAM, proporcionando sistemas completos y eficientes para la gestión y monitorización de la ejecución de procesos de negocio.

La estrategia utiliza la aproximación Model Driven Engineering –MDE-, en la que se representan de forma abstracta los conocimientos que pertenecen a un dominio mediante modelos de patrones de diseño que se repiten en él.

Para ello, define el concepto de acontecimiento basado en protocolos, como concreción del concepto general de acontecimiento, en donde los tres elementos (ep, o , d) que componen la tupla que lo define son: (1) el protocolo que se ejecuta –guía-, (2) el objeto sobre el que se ejecuta el protocolo –estructura- y (3), el efecto que produce dicha ejecución en el objeto –comportamiento-.

El efecto de un acontecimiento basado en protocolos puede ser de dos naturalezas diferentes: un cambio de estado del objeto o un cambio en los datos que existen asociados a los objetos.

Los datos asociados a los objetos que son efecto de una ejecución de protocolos son datos longitudinales, ya que están sujetos a dicha ejecución, la cual está localizada en el tiempo.

Sobre la base del concepto de acontecimiento basado en protocolos, se definen los patrones de diseño, patrones de acontecimientos, que permiten representar los distintos efectos de una ejecución de protocolo.

En la estrategia se definen cinco patrones de acontecimientos:

- Patrón de Acontecimientos de Cambios de Estado.
- Patrón de Acontecimientos de Creación de Objeto.
- Patrón de Acontecimientos de Registro de Datos de Objeto.
- Patrón de Acontecimientos de Cambio de Datos de Objeto.
- Patrón de Acontecimientos de Cambio de Datos de Estado.

En los patrones de acontecimientos se utiliza un perfil UML, el perfil de acontecimiento, que permite modelar los conceptos incluidos en ellos extendiendo la sintaxis UML al lenguaje de nuestro dominio.

Utilizando los patrones de acontecimientos anteriores, la estrategia de diseño define una metodología para la construcción de las bases de acontecimientos.

La metodología está basada en la aproximación MDA –Model Driven Architecture-, que consiste en la definición de dos modelos, el modelo PIM –Platform Independent Model- y el modelo PSM –Platform Specific Model-.

En la metodología, los patrones de acontecimientos se utilizan a modo de bloques de

construcción repetitivos para obtener el modelo PIM a desarrollar en la construcción de los SGBAP.

La metodología desarrolla el patrón PSM transformando el patrón PIM a las particularidades de las bases de datos relacionales, plataforma elegida en la estrategia de diseño para la implementación de las bases de acontecimientos.

Por otro lado, la estrategia de diseño desarrolla la arquitectura de un SGBAP basándose en el patrón MVC –Model View Controller-. El patrón MVC es un patrón de arquitectura de software que separa la gestión de datos, la lógica de negocio y la interfaz de usuario en una aplicación de software.

La arquitectura de un SGBAP, al basarse en el patrón MVC, permite un desarrollo modular de los protocolos que se implementan en él. Cada protocolo tiene sus clases de vista, controlador y modelo.

El sistema resultante con la arquitectura del SGBAP, es un sistema escalable y de fácil mantenimiento. Añadir un nuevo protocolo, o modificar uno existente, se realiza gestionando sus propias componentes sin afectar al resto de protocolos.

El uso de patrones de diseño, tanto para la obtención de la base de acontecimientos como para la construcción del sistema SGBAP, permite aprovecharse de los beneficios que proporciona esta forma de trabajo, como la obtención de elementos reutilizables en el diseño, mayor facilidad de aprendizaje y comunicación entre los diseñadores al disponer de un vocabulario y conocimiento establecido y común, o la estandarización del diseño dentro del entorno de aplicación de los patrones.

La arquitectura SGBAP, debido a la modularidad establecida en su diseño, permite la construcción del sistema como sistema distribuido.

La alta modularidad de la arquitectura y de los modelos de la base de acontecimientos, sin embargo, provoca el inconveniente del volumen de elementos que se construyen y definen en el sistema.

En el caso de sistemas con muchos protocolos a implementar, como en el caso ARASIS, el hecho de que elemento de los modelos PIM y PSM -objetos, protocolos, estados, transiciones…- tengan su propia clase UML, traducidas en tablas en la base de datos relacional final, provoca que el tamaño final de estos modelos llegue a tener un tamaño considerable, lo que puede provocar una dificultad en su gestión.

En este caso de grandes sistemas, la transformación del modelo PIM al PSM puede resultar costosa, ya que las transformaciones se realizan manualmente. No obstante, la existencia de ambos modelos permite al diseñador mantener la consistencia entre el diseño conceptual del modelo PIM y el diseño implementado en la plataforma específica con el modelo PSM.

En el caso del código desarrollado del propio software, se produce la misma desventaja, la implementación de forma modular de los protocolos implica la existencia de múltiples ficheros de código para desarrollar la lógica de todos los protocolos.

La complejidad general del código, sin embargo, es muy reducida, ya que aunque existan muchos ficheros de código, cada protocolo tiene definidas sus clases de vista, modelo y controlador que se repiten en forma de bloque para cada uno de ellos.

Los sistemas OcTrail y de Business Intelligence proporcionan unos entornos muy completos

para la explotación total de toda la información almacenada en los acontecimientos.

El sistema OcTrail permite obtener de forma eficiente las historias y las líneas de vida de los objetos, dotando al sistema SGBAP de los mecanismos mínimos para ser considerado como tal. Permite, por tanto, obtener el *provenance* de los datos.

El sistema de Business Intelligence utiliza la tecnología OLAP para el análisis dinámico de la información almacenada en la base de acontecimientos. La tecnología OLAP proporciona un método eficiente de explotación de grandes volúmenes de datos

Por otro lado, la estrategia de diseño proporciona los mecanismos necesarios para afrontar la evolución de protocolos que pueden producirse por cambios en los procesos de negocio.

En este caso, la estrategia aboga por el versionado de protocolos, como mecanismo de preservación del *provenance* de los datos, pero también desarrolla otro tipo de mecanismos menos complejos: el cambio de protocolo y la evolución de protocolo.

La solución del versionado de protocolo permite conservar tanto la información de todos los acontecimientos generados en el tiempo, como cuál ha sido su origen –*provenance*-, es decir, con qué protocolo exacto se han registrado.

Los mecanismos de cambio de protocolo y de evolución de protocolo, sin embargo, tienen el inconveniente de que se produce una pérdida en la información almacenada en la base de acontecimientos, o en el conocimiento de cómo se ha generado la información.

La estrategia anterior se ha probado con éxito en la prueba de concepto de ARASIS, donde se ha implementado un sistema para la gestión de los protocolos que se realizan en un biobanco de muestras biológicas. El sistema ARASIS es una herramienta eficiente y donde se gestiona toda la información que se requiere en un contexto tan complejo como la investigación biomédica.

La estrategia también se ha utilizado para construir sistemas reales en otros contextos, como para la gestión de incidencias siguiendo las buenas prácticas de ITIL v3.

Por último, la estrategia de diseño comentada anteriormente define una forma concreta para el diseño y construcción de sistemas gestores de bases de acontecimientos basados en protocolos –SGBAP-, pudiendo existir otras estrategias alternativas.

Como muestra, se ha expuesto una alternativa de diseño que se está probando actualmente, en la que se modifica los patrones de acontecimientos que permiten obtener los modelos PIM y PSM en la metodología de las bases de acontecimientos.

Patrón para la evaluación del desempeño de los recursos de una empresa

Los métodos de asignación de tareas normalmente consideran una foto fija de la actualidad de los costes para realizar la asignación, en base a las capacidades técnicas, costes monetarios de los recursos…, pero no utilizan un aspecto fundamental como es la historia de cómo han desempeñado los recursos las tareas en el pasado.

El Patrón de Acontecimientos basados en Tareas nos proporciona dos grandes beneficios:

- Una estructura para el almacenamiento de la historia de ejecución de las tareas por parte de los recursos, almacenando los acontecimientos que se generan por dichas ejecuciones.

- La información necesaria para poder definir indicadores de rendimiento de los recursos utilizando los acontecimientos almacenados, desde el comienzo de desempeño del recurso hasta la actualidad.

El desarrollo del patrón requiere la concreción del concepto de acontecimiento al contexto de la gestión de tareas, el acontecimiento basado en tareas.

En este acontecimiento, a diferencia del acontecimiento basado en protocolos, el núcleo del acontecimiento son los recursos, los actores que realizan las acciones o tareas, y no los objetos sobre los que se realizan.

En el acontecimiento basado en tareas, por tanto, la tupla de tres elementos (t, o d) que componen su definición son: (1) la tarea que se ejecuta –guía-, (2) el objeto que realiza la tarea –estructura- y (3), el efecto que produce dicha ejecución –comportamiento-.

El patrón permite además la detección de comportamientos o desempeños deficientes por parte de los recursos. Asociados a los indicadores de rendimiento se pueden definir límites que en caso de incumplimiento avisen de estas deficiencias.

La estructura definida en el patrón puede aplicarse a contextos de muy diferente naturaleza. Esto ha quedado demostrado en la aplicación del patrón en las pruebas de concepto de una atención al cliente, un sistema de salud y un sistema de vigilancia de robots.

El principal inconveniente, al igual que en los acontecimientos basados en protocolos, es la necesidad de capacidad de procesamiento y almacenamiento que requiere la gestión de todos los acontecimientos que se generan en la ejecución de las tareas. Sin embargo, como se ha visto en la aplicación del caso ARASIS, esta capacidad crece linealmente con el volumen de acontecimientos, manteniéndose en márgenes eficientes.

El patrón nos proporciona, a través de los indicadores de rendimiento que se pueden definir sobre los acontecimientos, un mecanismo complementario a los métodos de optimización habituales, como los métodos de programación lineal o método Húngaro, utilizando la historia del desempeño para realizar una mejor asignación de los recursos a las tareas.

La mejora se consigue mediante la modificación de los pesos que se utilizan en las funciones de optimización, aplicando un factor de ajuste basado en los indicadores de rendimiento calculados.

Por otro lado, la asignación de tareas también puede realizarse utilizando exclusivamente los indicadores de rendimiento, solucionando aquellos casos en los que no se puede identificar una variable a optimizar, o esta variable es igual para todos los recursos.

10.1 Publicaciones

A lo largo del desarrollo de la tesis, se han publicado varios artículos que o bien describen los conceptos investigados o los utilizan para desarrollar nuevos conceptos y teorías dentro de la línea de investigación del grupo Nóesis:

- Occurrence-Oriented Design Strategy for Developing Business Process Monitoring Systems

  Autores: Eladio Domínguez, Beatriz Pérez, Ángel L. Rubio, María A. Zapata, Alberto

Allué, Juan Lavilla

Revista: IEEE Transactions on Knowledge and Data Engineering

ISBN: 1041-4347 (ISNN)

Editorial: IEEE Computer Society

Lugar de publicación: DOI: 10.1109/TKDE.2013.166

Artículo publicado en el año 2014 en revista indexada en el JCR con factor de impacto en el tercio superior, en el que se describe una versión inicial de la estrategia de diseño.

- The task-oriented occurrence pattern

  Autores: Alberto Allué, José Carlos Ciria, Eladio Domínguez, Ángel Francés, Antonio López, María A. Zapata

  Publicado en: Proceeding EuroPlop '16 Proceedings of the 21st European Conference on Pattern Languages of Programs (p. 6)

  Editorial: ACM

  DOI: 10.1145/3011784.3011790

  Artículo publicado en el congreso EuroPLoP de julio de 2016, donde se describen los conceptos del patrón *Patrón de Acontecimientos basados en Tareas* para la evaluación de desempeño basado en la historia.

- Developing provenance-aware query systems: an occurrence-centric approach

  Autores: Eladio Domínguez, Beatriz Pérez, Ángel Luis Rubio, María A. Zapata, Alberto Allué, Antonio López

  Revista: Knowledge and Information Systems 50, pp 661–688, 2017

  ISBN: 0219-1377 (ISNN)

  Editorial: Springer London

  Lugar de publicación: DOI: 10.1007/s10115-016-0950-z

  Artículo publicado en revista indexada en el JCR con factor de impacto en el tercio superior, en el que se describe un lenguaje de consultas de acontecimientos sobre una base de acontecimientos que se desarrolla mediante la estrategia de diseño.

- QRP: a CMMI Appraisal Tool for Project Quality Management

  Autores: Alberto Allué, Eladio Domínguez, Antonio López, María A. Zapata

  Revista: Knowledge and Information Systems

  Presentado en el congreso PRojMAN 2013 - International Conference on Project MANagement

  Publicación: Procedia Technology

  ISBN: 2212-0173 (ISSN)

  Volumen: 9, págs. 664 - 669

  Editorial: Elsevier

  Artículo presentado en el congreso PRojMAN de 2013, en el que se describe una aplicación de gestión de calidad siguiendo el modelo CMMI, en cuya base de datos se utilizan las estructuras longitudinales primitivas asociadas a los procesos de negocio de CMMI que dan origen al concepto de acontecimiento, y sobre las que se

implementa la tecnología OLAP incluida en la arquitectura de los sistemas SGBAP de la tesis.

- Generating persistence structures for business process monitoring purposes: the Occurrence–centric approach

  Autores: Eladio Domínguez, Beatriz Pérez, Ángel L. Rubio, María A. Zapata, Alberto Allué, Antonio López

  Artículo enviado al 15th International Conference on Business Process Management (BPM 2017) con una propuesta que promueve la generación automática de estructuras de almacenamiento para la monitorización utilizando el concepto de acontecimiento y la estrategia de diseño.

10.2 Líneas de trabajo futuro

Las líneas de trabajo futuras tras el desarrollo de esta tesis se centran en dos vertientes. Por un lado, en la continuación de la mejora y avance de la teoría de acontecimientos y la estrategia de diseño desarrollada. Por otro lado, en la exploración e investigación del uso de los acontecimientos en otros contextos diferentes a los procesos de negocio.

Líneas en la estrategia de diseño

La primera línea de trabajo consiste en seguir optimizando y mejorando en la propia estrategia de diseño para la construcción de SGBAPs. En esta línea se enmarca la alternativa de estrategia presentada en el Capítulo 8, en la que se han introducido mejoras para el acceso a los datos de tiempo en el modelo de la base de acontecimientos, y que se está probando y aplicando en los nuevos sistemas que se están construyendo actualmente. Otro de los puntos a mejorar en la estrategia es la transformación del modelo PIM al modelo PSM. Actualmente, esta transformación se realiza de forma manual, lo que puede llegar a ser costoso en sistemas de gran alcance como ARASIS. El objetivo es lograr la implementación de mecanismos que permitan realizar esta transformación de una forma automática.

Por otra parte, en la tesis doctoral 'Lenguaje de consultas para la gestión de acontecimientos' de Antonio López Martínez (Universidad de Alcalá, 2017), se ha desarrollado un lenguaje para la gestión y consulta de los acontecimientos que se almacenan en una base de acontecimientos construida con la estrategia de diseño desarrollada en esta tesis, a semejanza del lenguaje SQL en las bases de datos convencionales. Una versión inicial de este lenguaje se ha publicado en el artículo [31], del que el autor de esta tesis es coautor, en la revista *Knowledge and Information Systems* indexada en el JCR. Una gran línea de trabajo futura es modificar los elementos necesarios de la estrategia actual para incluir la gestión de los acontecimientos a través de ese lenguaje, permitiendo consultar y gestionar los acontecimientos en un lenguaje propio y adaptado a las estructuras de acontecimientos. Esto nos proporcionaría una gran mejora de la eficiencia en la obtención de las estructuras de acontecimientos complejas, como son la historia y la línea de vida de los objetos.

Respecto al patrón para la evaluación del desempeño, la principal línea de trabajo consiste en el estudio de la aplicación de los indicadores a métodos de optimización existentes, con el objetivo de demostrar que la utilización de la historia obtenida con nuestro modelo mejora la

asignación de tareas.

Por otro lado, el acontecimiento basado en protocolos y el acontecimiento basado en tareas proporcionan dos enfoques diferentes en la explotación de un conocimiento que puede ser común, la ejecución de unas determinadas acciones por unos actores que generan unos resultados. En la versión de protocolos se focaliza en el efecto de las acciones, mientras que en la versión de tareas se focaliza en los actores para evaluar su desempeño. Sin embargo, en ocasiones es posible que se necesite un sistema en el que se pueda analizar la información en base a los dos enfoques al mismo tiempo. El objetivo en el futuro sería unir ambos enfoques en una estructura común, desarrollando la estrategia para implementar sistemas de monitorización que permita analizar tanto la información de los procesos como el desempeño de los recursos.

Por último, el concepto de acontecimiento que se ha utilizado en la tesis, tal y como se ha explicado en el Capítulo 3, se centra en aquellos acontecimientos artificiales en el que se puede identificar de forma concreta el objeto y los pasos o tareas que existen en la acción que acontece. En esta línea, existe otra gran línea de trabajo para investigar en otro tipo de acontecimientos, como los acontecimientos por fenómenos naturales, desarrollando sistemas y estructuras que permitan reflejarlos en sistemas de información.

Utilización en otros campos de investigación

Además de las líneas de trabajo sobre la propia estrategia, otra vía de trabajo es la investigación del uso de los acontecimientos en otros campos de investigación en auge. En esta línea se encuentra la investigación en el ámbito del Internet of Things -IoT- [62][9]. Los acontecimientos nos proporcionan una forma completa y eficiente para la gestión de la historia de los objetos, lo que puede ser muy útil para problemas actuales de investigación como la autenticación de los objetos en el IoT para evitar suplantaciones de identidad. La historia de un objeto puede ser utilizada para elaborar un mecanismo de autenticación basada en la propia historia del objeto, realizándole preguntas que sólo el objeto puede conocer sobre acontecimientos que se han generado por su actuación en el pasado.

De la misma forma, en el ámbito de la robótica los acontecimientos pueden ser una solución eficiente para almacenar el conocimiento de los robots, así como una vía para desarrollar sistemas de seguridad que utilicen los robots como agentes de seguridad y en la que los hechos de seguridad se gestionen como acontecimientos.

En estas líneas de investigación se realizaron los proyectos THOFU [84] o SMOTY [82], donde se iniciaron investigaciones sobre sistemas de seguridad sobre la base de la teoría de acontecimientos, pero sin duda es una gran línea de investigación futura.

VI. APÉNDICES

Apéndice A. Entornos desarrollados en el SGBAP de ARASIS

En este apéndice se explican con más detalle los principales entornos que se han desarrollado en el SGBAP de ARASIS.

A.1 Entorno dinámico del OcTrail Graphical Manager

El entorno construido en el OcTrail Graphical Manager es un entorno dinámico a modo de cuadro de mandos que permite visualizar al mismo tiempo varias líneas de vida, organizándolas en una interfaz que permite definir 'pestañas de componentes gráficos':

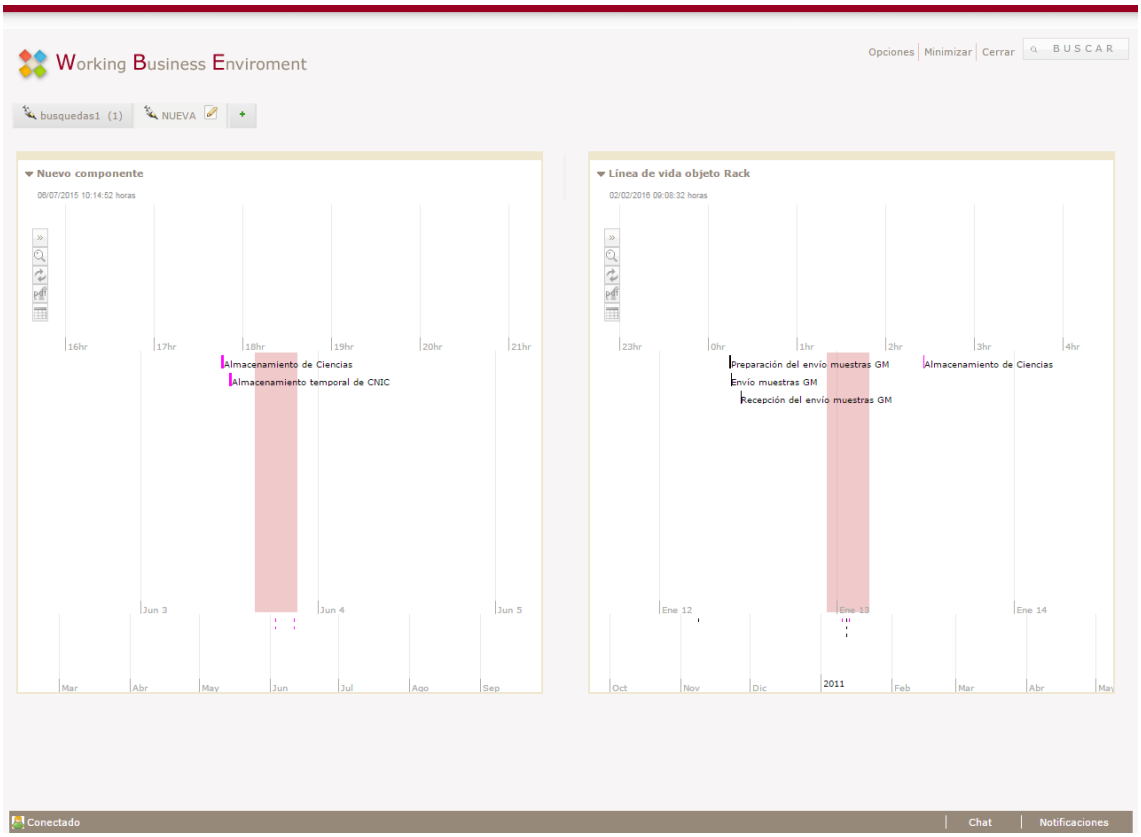


Figura A-1 Interfaz OcTrail Graphical Manager en ARASIS

En el entorno ejemplo de la imagen se pueden ver dos pestañas, teniendo en la pestaña actual de trabajo dos componentes en las que se muestran dos líneas de vida diferentes al mismo tiempo.

Estas pestañas y componentes se pueden personalizar, cambiándoles el nombre, los colores de los bordes, se pueden intercambiar componentes entre las pestañas…, de forma que sea más sencilla la organización de las líneas de vida consultadas.

Toda esta personalización y organización se guarda en un fichero de configuración XML, donde para cada usuario se almacena qué pestañas y componentes tiene en su entorno de trabajo, con las características que haya personalizado, y con las condiciones de filtro que han permitido generar las líneas de vida, de forma que cuando vuelva a iniciar el entorno del sistema OcTrail se le carguen de nuevo las mismas líneas de vida que visualizó la última vez, y configuradas y organizadas exactamente igual como las dejó en la última sesión.

Las líneas de vida en los componentes son interactivas para moverse en el tiempo gracias al widget proporcionado por la librería Simile Timeline, que explicaremos en detalle en el apartado Apéndice B.

A través de los acontecimientos que se visualizan en las líneas de tiempo podemos obtener más información acerca de los protocolos que se han ejecutado:

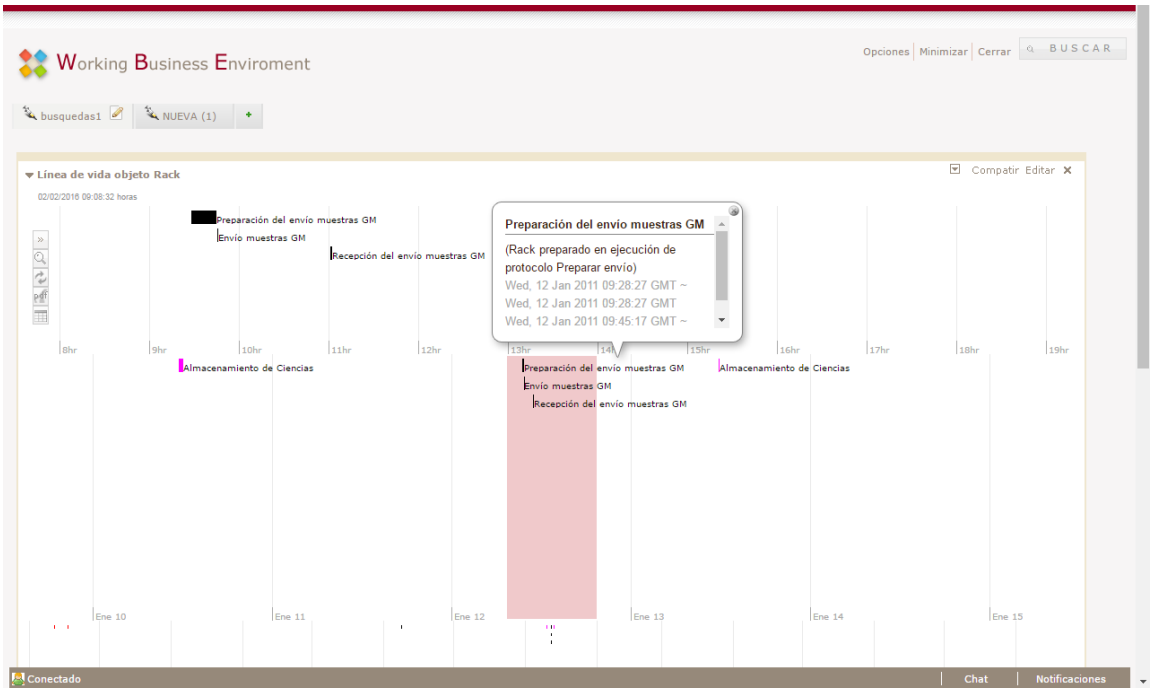


Figura A-2 Interfaz OcTrail Graphical Manager - Burbuja

Haciendo clic sobre un acontecimiento se nos muestra una burbuja con la información detallada del acontecimiento, con las fechas de cuándo se ha producido y acceso al detalle del protocolo ejecutado.

Por último, los acontecimientos de las líneas de vida, además del formato gráfico proporcionado por la librería Simile Timeline, también se pueden visualizar en un listado de

acontecimientos en formato tabular, tal como se muestra en la siguiente imagen:

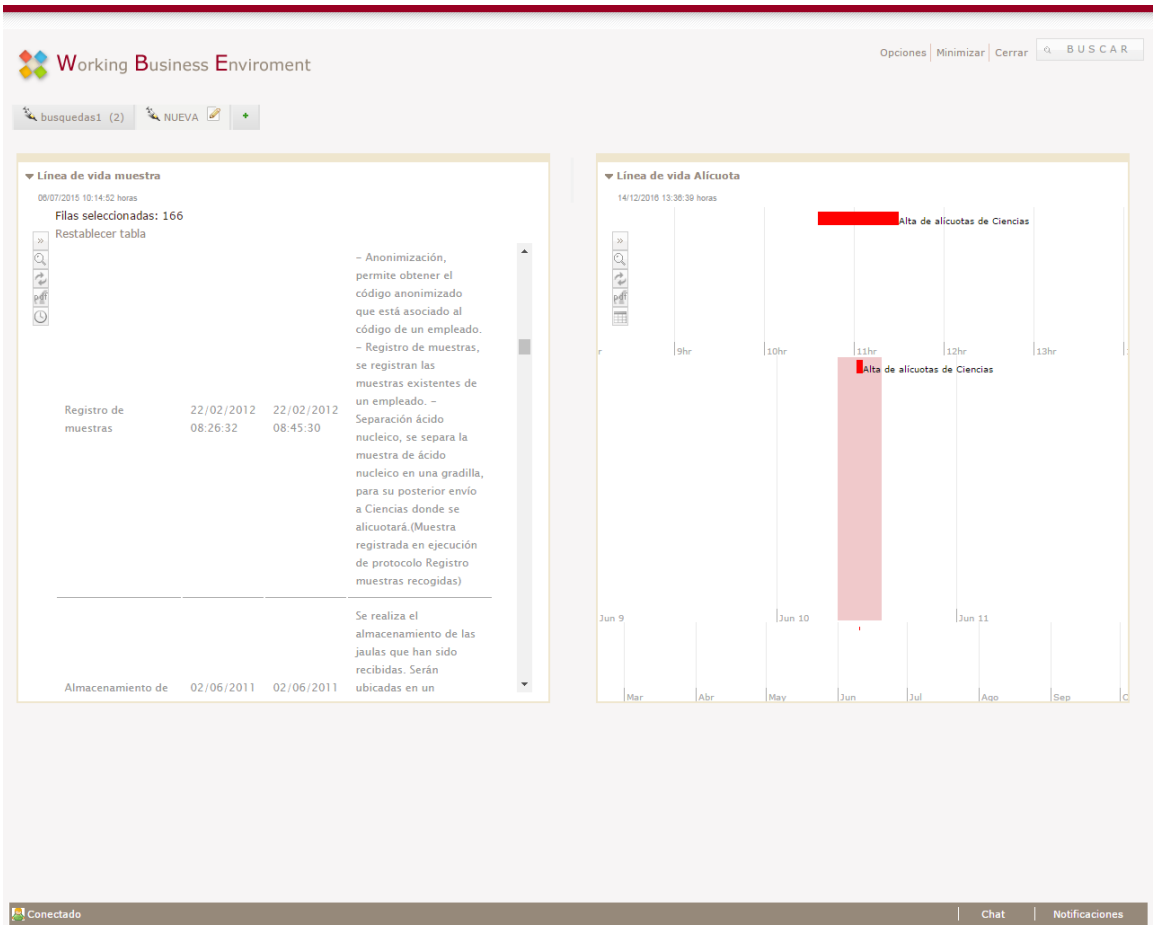


Figura A-3 Interfaz del OcTrail Graphical Manager en ARASIS - Tabla

A.2 Red de líneas de vida –historia de los objetos- en el sistema OCTrail

Uno de los mecanismos para explotar el conocimiento de la OcBase que debe proporcionar todo SGBAP, tal y como se definió en el apartado 3.3, es una red de líneas de vida que permita consultar la historia de los objetos.

Esta premisa se cumple en el sistema OcTrail de ARASIS. Cuando se visualiza una línea de vida de un objeto concreto, se tiene acceso a todos los objetos relacionados en su historia, pudiendo acceder a la línea de vida de estos y que forman parte de forma indirecta de la historia del objeto original.

El ejemplo más representativo del concepto de historia de objeto es el del empleado. La línea de vida del empleado consistirá en los acontecimientos generados en la ejecución de los protocolos que han actuado directamente sobre él, que si recordamos eran los del registro del

consentimiento. Sin embargo, como parte de su historia se encuentran todos los acontecimientos que se produzcan sobre las muestras que se le han extraído y que se gestionan en el biobanco.

En las siguientes imágenes se ve el ejemplo respecto a una muestra.

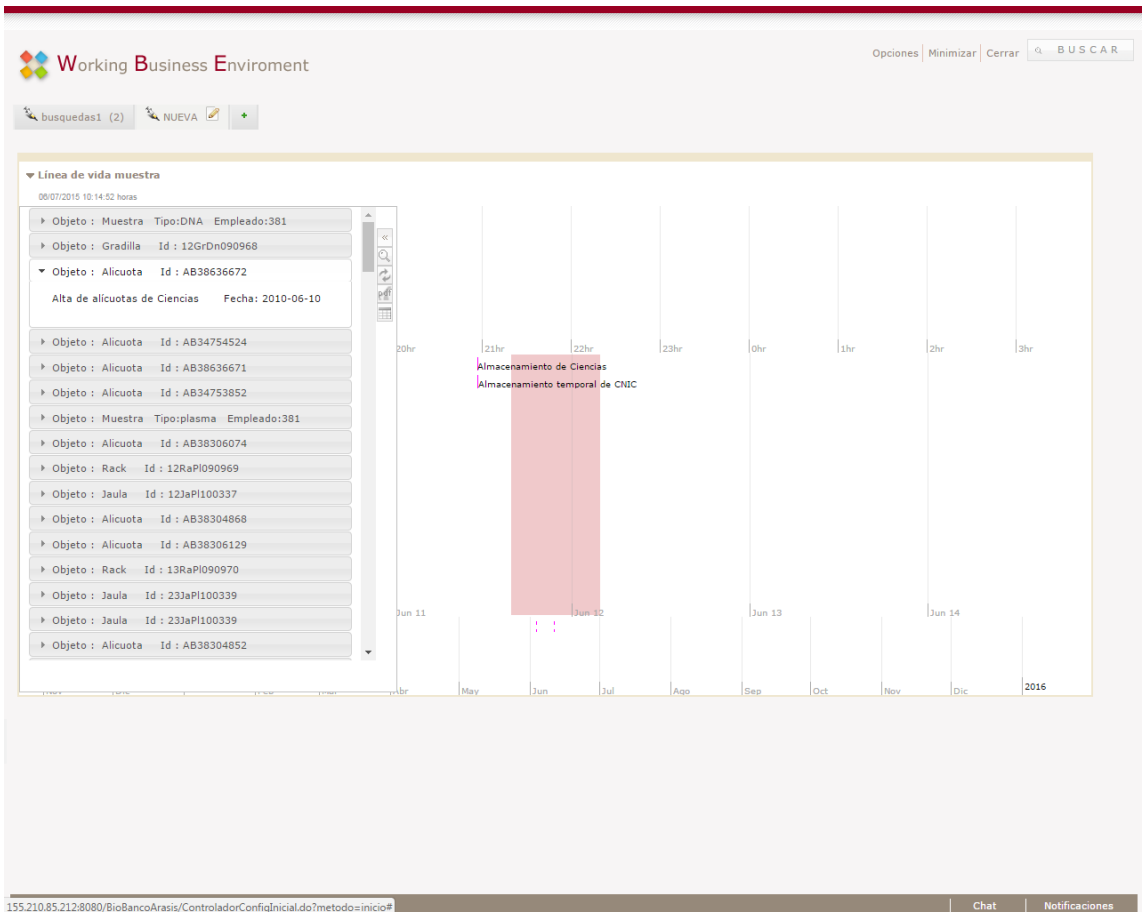


Figura A-4 Red líneas de vida en ARASIS - Búsqueda

En la imagen anterior estamos visualizando la línea de vida de la muestra, pero tenemos acceso a los objetos relacionados de su historia, listados en la parte izquierda de la imagen, como la alícuota resaltada.

Al hacer doble clic sobre la alícuota, automáticamente se nos añade otra componente con la línea de vida de esa alícuota en cuestión, pudiendo repetir el proceso las veces que necesitemos:

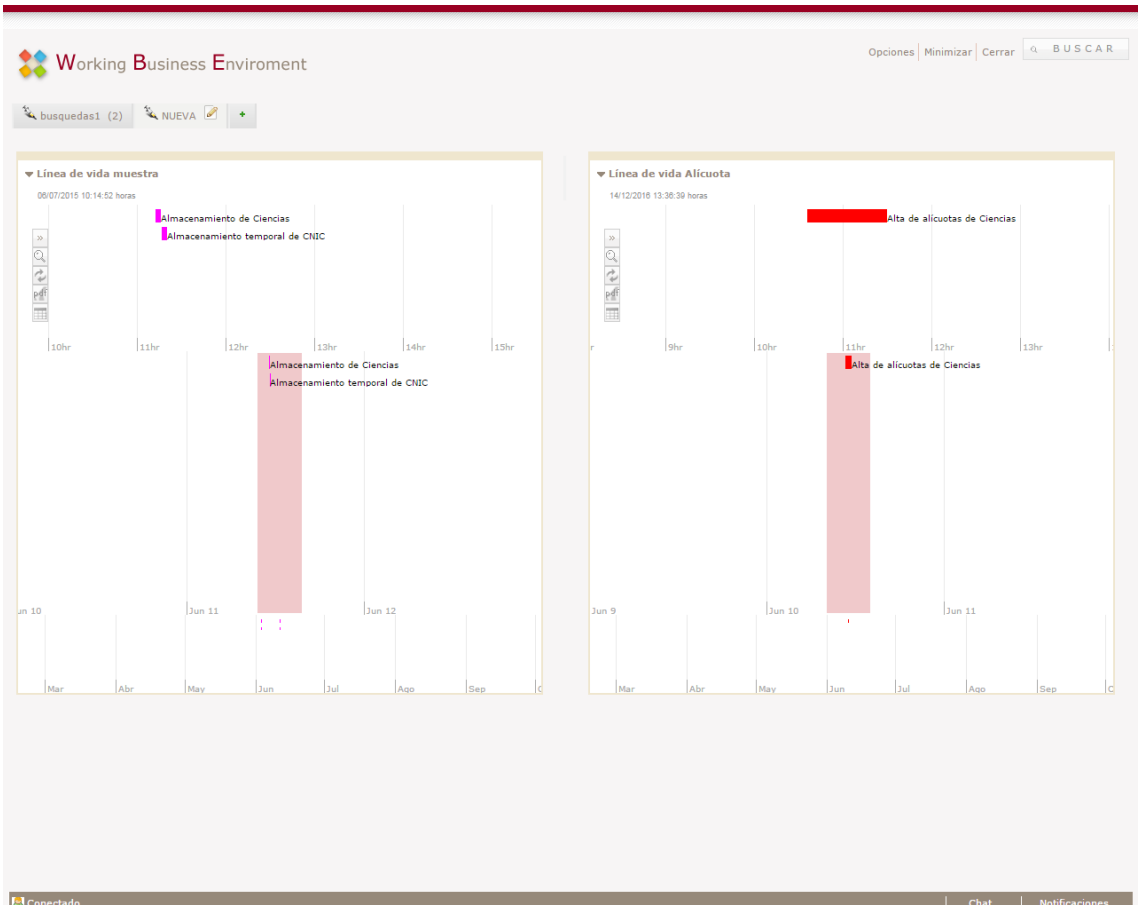


Figura A-5 Red líneas de vida en ARASIS - Resultado

En este pequeño ejemplo se puede ver cómo se utiliza esa red de líneas de vida. De esta forma, a través del sistema OcTrail de ARASIS, podemos explotar las historias y las líneas de vida de los objetos.

A.3 Líneas longitudinales de datos en el sistema OcTrail

En las líneas de vida comentadas en el apartado anterior del sistema OcTrail de ARASIS, se representan los acontecimientos basados en protocolos cuyo efecto son cambios de estado de los objetos.

Sin embargo, si recordamos en ARASIS también se recogen acontecimientos de protocolos cuyo efecto son el registro de datos longitudinales asociados al objeto empleado, como los análisis clínicos, los Cuadernos de Recogida de Datos –CRD- y los formularios de salud.

En el sistema OcTrail de ARASIS, se pueden generar también las llamadas 'líneas longitudinales de datos clínicos' para explotar la información de estos datos longitudinales. En este caso, lo que se representan en las componentes del 'OcTrail Graphical Manager' son líneas de tiempo, similares a las líneas de vida, en las que se muestran longitudinalmente todos los análisis, CRDs y formularios registrados en el sistema para un empleado a lo largo de un periodo de tiempo, mediante la representación de los acontecimientos generados por la

ejecución de los protocolos de datos correspondientes.

Al igual que con las líneas de vida, la búsqueda de estos acontecimientos puede filtrarse a través de un conjunto de campos de búsqueda definidos en el OcTrail Query Manager, en el que se pueden establecer como parámetros de búsqueda datos concretos contenidos en estos datos longitudinales.

En la siguiente imagen se puede ver el filtro de datos clínicos para realizar una consulta de este tipo en el OcTrail Query Manager:

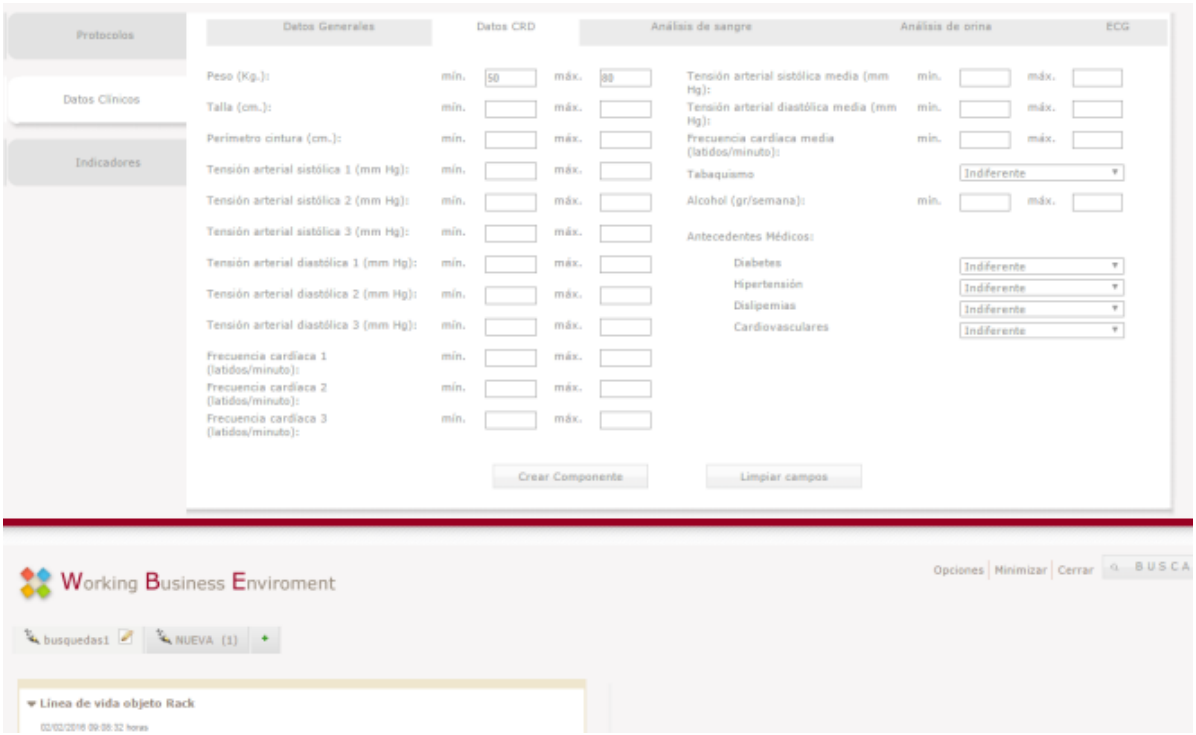


Figura A-6 Filtro de datos clínicos en OcTrail Query Manager de ARASIS

Como se puede ver en la imagen anterior, el filtro se compone de los datos concretos de cada dato longitudinal registrado, en los que se pueden establecer rangos de búsqueda o seleccionar opciones dependiendo de la naturaleza del dato. El resultado de la búsqueda son los empleados que tienen datos longitudinales que cumplan todas las condiciones de búsqueda establecidas.

Esta búsqueda la realizan los componentes del OcTrail Retrieval Manager de la capa de negocio, en la que está implementado un sistema de consultas dinámicas contra la OcBase para consultar la información de estos acontecimientos según las condiciones establecidas.

En la siguiente imagen puede verse un ejemplo de línea longitudinal de datos del sistema OcTrail, en donde se muestran todos los acontecimientos registrados sobre datos longitudinales para el empleado seleccionado por el usuario (resultado de una búsqueda realizada en el OcTrail Query Manager).

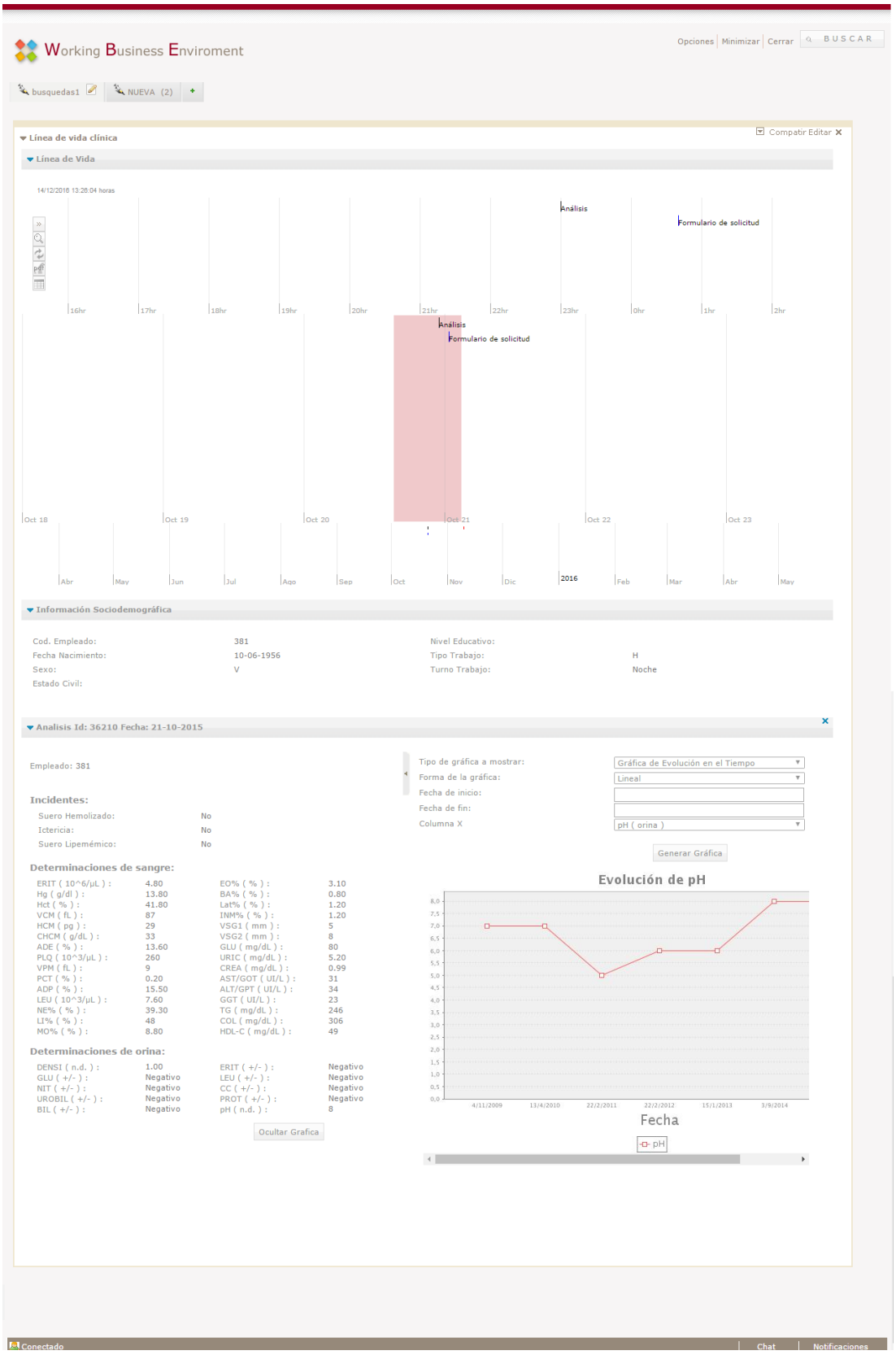


Figura A-7 Ejemplo de Línea longitudinal de datos

En la interfaz de estas líneas de tiempo puede explotarse toda la información registrada de los datos longitudinales. Como puede verse en la imagen, en la línea de tiempo se muestran todos los acontecimientos existentes (a través de los protocolos de registro de datos) en el periodo de tiempo establecido en la búsqueda. Al seleccionar un acontecimiento en la línea, en la parte inferior se muestra toda la información actualizada del dato longitudinal que se registró como efecto del protocolo, con acceso a la información de la posible evolución de un dato concreto en el tiempo. De esta forma, en el sistema OcTrail se obtiene una explotación exhaustiva de todos los acontecimientos registrados en la OcBase.

A.4 Entorno de análisis dinámico de datos del sistema BI

El entorno de análisis de datos nos permite consultar los cubos OLAP construidos de forma libre mediante las funcionalidades que nos proporciona la librería JPivot. En la siguiente imagen puede verse la interfaz de este entorno:

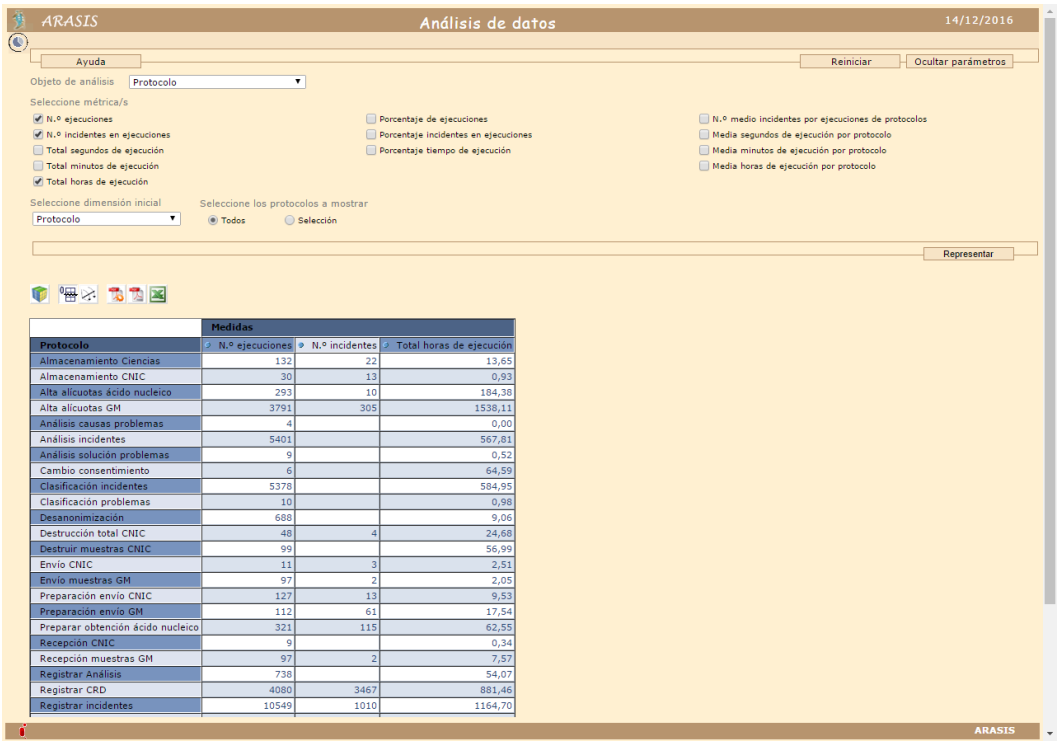


Figura A-8 Interfaz análisis de datos Sistema BI en ARASIS

En la imagen podemos ver las dos componentes de la capa de presentación:

- BI Query Manager del entorno, en la parte superior. Nos permite seleccionar el cubo OLAP con el que se quiere trabajar, así como las métricas y dimensiones iniciales que van a permitir construir la MDX inicial –en el controlador de gestión del entorno- para iniciar el funcionamiento de la librería JPivot en el BI Graphical Manager.

  El cubo OLAP se selecciona especificando el objeto de análisis, donde se pueden seleccionar los objetos del sistema de la OcBase –empleado, muestra, alícuota…- y

otros objetos de análisis adicionales que son importantes para la explotación del conocimiento de la OcBase. Por ejemplo, el cubo OLAP centrado en las ejecuciones de protocolos que vemos en la imagen anterior, o los cubos OLAP referentes a los incidentes gestionados, los datos clínicos registrados sobre los empleados.

Una vez seleccionado el cubo OLAP, en el BI Query Manager se muestran las métricas y las dimensiones que tiene definidas el cubo para realizar la selección inicial mencionada.

- BI Graphical Manager del entorno, en la parte inferior. Interfaz que proporciona la librería JPivot (con la personalización realizada para el sistema ARASIS), donde se muestran los resultados de las consultas MDX realizadas contra el cubo OLAP en un formato tabular y se tiene acceso a los iconos de trabajo sobre el análisis.

  Entre las funcionalidades se encuentran: (1) la navegación entre los diferentes niveles de los miembros de las dimensiones –mediante los iconos de + y – en la tabla de resultados, (2) modificar el análisis –icono del cubo- para cambiar las métricas y dimensiones que se muestran en el análisis, (3) añadir filtros y condiciones a las consultas (por ejemplo, establecer que cualquier resultado sea para un año en concreto), (4) generación de gráficos y (5) opciones de exportación de los resultados en PDF o Excel.

  En la siguiente imagen se muestra un ejemplo en el que el análisis realizado en la imagen anterior, se ha modificado para quitar métricas, añadir la dimensión de Fecha al análisis y descender por sus datos hasta el nivel de año para obtener el desglose de resultados.

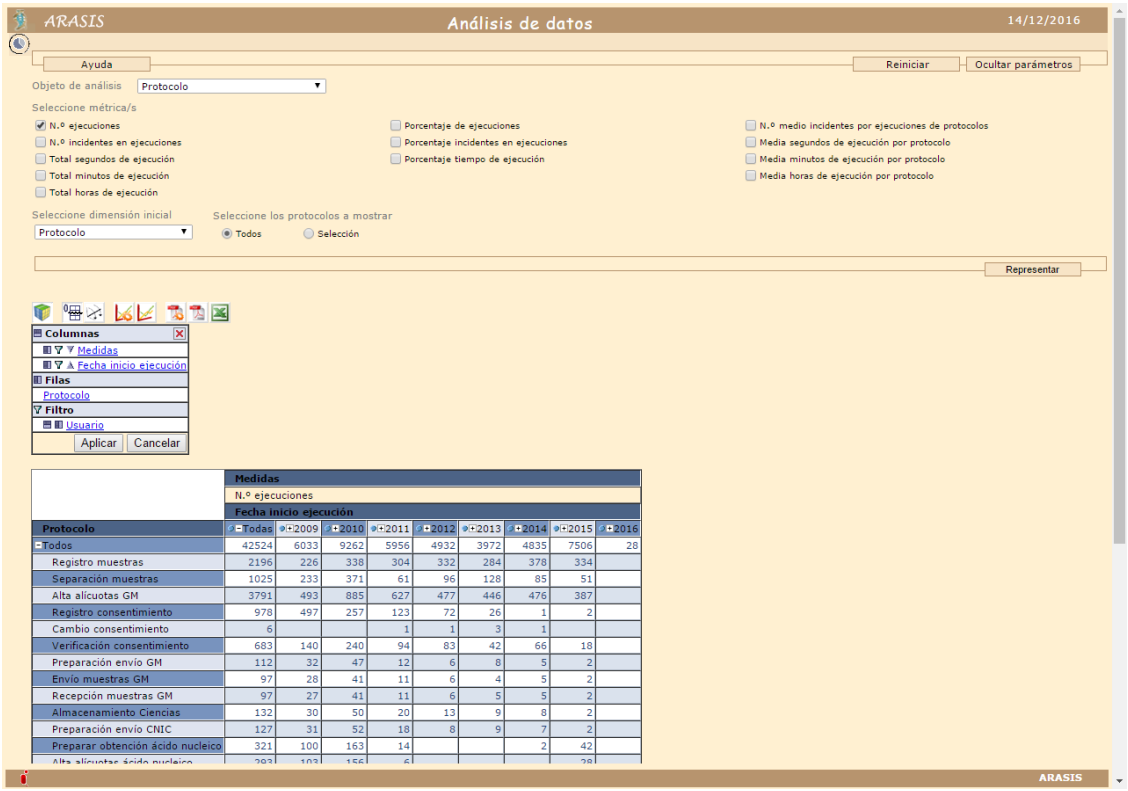


Figura A-9 Interfaz análisis de datos Sistema BI en ARASIS – Modificación dinámica

A.5 Entorno de indicadores longitudinales del sistema BI

Un indicador longitudinal nos permite obtener la evolución de un dato a lo largo del tiempo.

El entorno de indicadores longitudinales permite consultar los cubos OLAP mediante una serie de indicadores longitudinales predefinidos según las necesidades del estudio del proyecto AWHS, como la evolución de pesos, índices de masa corporales, evolución de consentimientos de empleados…

En la siguiente imagen puede verse la interfaz de este entorno:

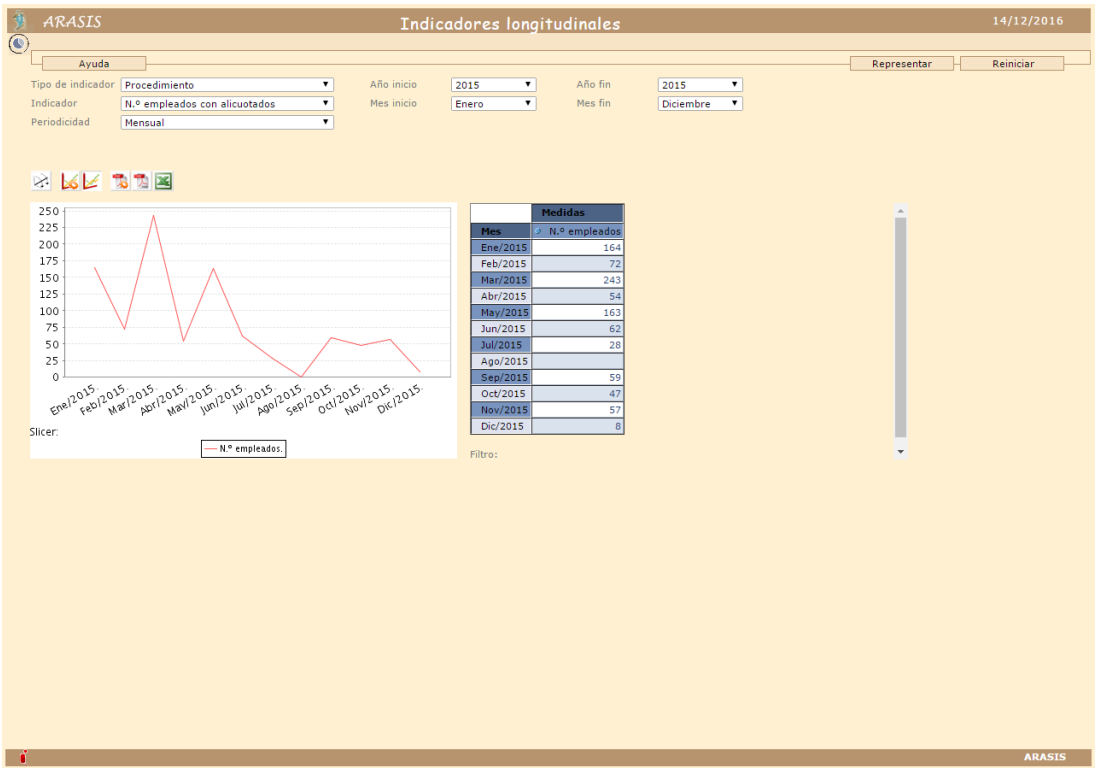


Figura A-10 Interfaz indicadores longitudinales Sistema BI en ARASIS

En la interfaz podemos ver las dos componentes de la capa de presentación:

- BI Query Manager, en la zona superior. Permite seleccionar el indicador entre todos los indicadores posibles, indicar la periodicidad en la que queremos obtener el resultado (diario, mensual o anual) y el periodo en el cual se quiere obtener el indicador. Esta selección permitirá construir la consulta MDX inicial –en el controlador de gestión del entorno- para iniciar el funcionamiento de la librería JPivot en el BI Graphical Manager.

  En el entorno existen 40 indicadores longitudinales diferentes, tanto de procedimiento (sumas, medias y porcentajes de empleados con alicuotados, transportes de muestras, incidentes…) como de datos clínicos (medias y porcentajes de pesos, tabaquismo, alturas, tensiones…).
- BI Graphical Manager, en la parte inferior. El indicador se representa con la Interfaz que proporciona la librería JPivot, al igual que el resto de entornos BI.

En este caso, se muestra tanto la tabla de resultados del cubo OLAP como un gráfico donde se muestra la evolución del indicador, y las funcionalidades de la librería se restringen a la modificación del gráfico para cambiar y personalizar su aspecto y la exportación de los datos a PDF y Excel.

A.6 Entorno de indicadores transversales del sistema BI

Un indicador transversal nos permite obtener el valor de un dato en un momento puntual del tiempo.

El entorno de indicadores transversales consulta los cubos OLAP, al igual que en el entorno de indicadores longitudinales, mediante un conjunto de indicadores transversales definidos para el estudio del proyecto AWHS.

En la siguiente imagen puede verse la interfaz de este entorno:

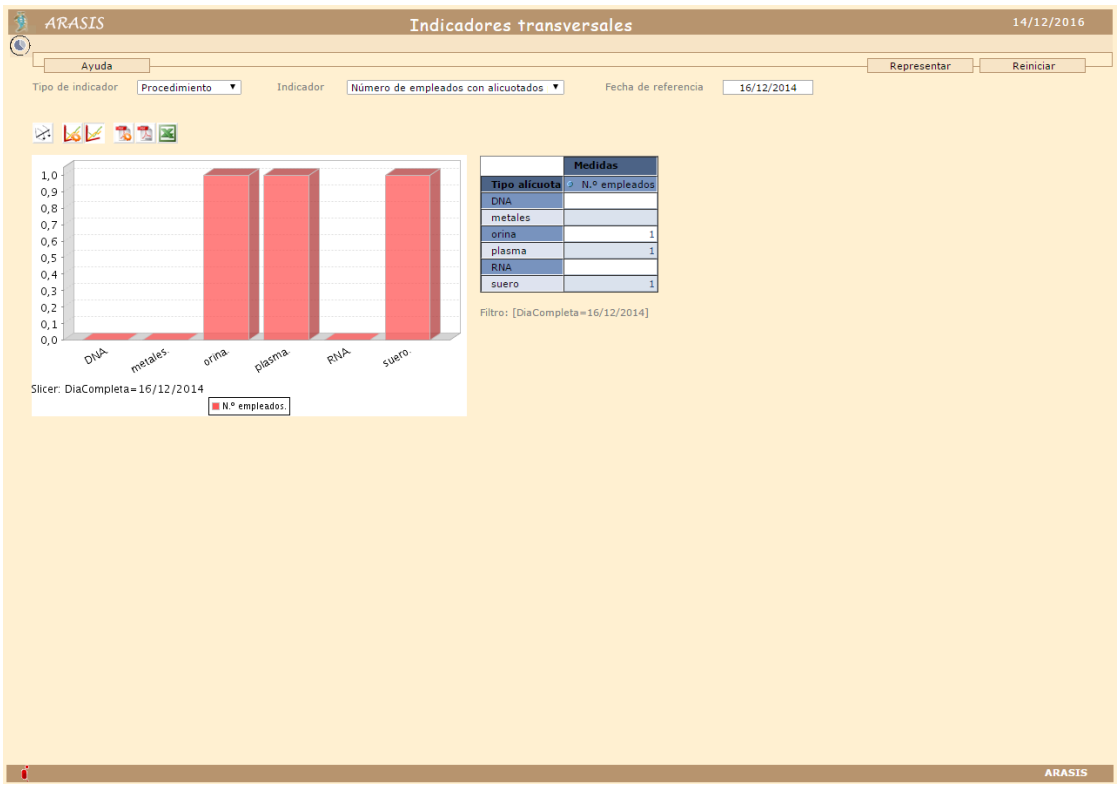


Figura A-11 Interfaz indicadores transversales Sistema BI en ARASIS

En la interfaz podemos ver las dos componentes de la capa de presentación:

- BI Query Manager, en la zona superior. Permite seleccionar el indicador y la fecha concreta en la cual se quiere obtener su valor para construir la consulta MDX inicial –en su correspondiente controlador de gestión- para iniciar el BI Graphical Manager.

  En este entorno se pueden consultar 25 indicadores transversales diferentes, y al igual que en los indicadores longitudinales, tanto de procedimiento (empleados con

alicuotados, transportes de muestras, incidentes…) como de datos clínicos (medias y porcentajes de pesos, tabaquismo, alturas, tensiones…).

- BI Graphical Manager, en la parte inferior. El indicador, como en el resto de entornos BI, se muestra con la interfaz desarrollada mediante la librería JPivot.

  La representación del indicador se realiza, como en los indicadores longitudinales, mediante la tabla de resultados del cubo OLAP y el gráfico que representa el indicador, estando restringidas las funcionalidades de la librería a la modificación del gráfico y a la exportación de los datos en formato PDF y Excel.

A.7 Entorno de perfil del biobanco del sistema BI

En el entorno de perfil de biobanco se puede obtener un perfil tanto del biobanco como de la cohorte de estudio –los empleados- en base a unas características que lo definen:

- Perfil del biobanco. Descripción del biobanco mostrando información del número de sujetos del estudio –empleados-, número de alícuotas, tipos de muestra –sangre, orina…- y protocolos diferentes que se aplican en el biobanco y porcentaje de ejecuciones según cada tipo de protocolo.
- Perfil de la cohorte. Permite obtener una descripción del sujeto de estudio medio de la cohorte, en base a su sexo, edad, índice de masa corporal y tabaquismo.

En la siguiente imagen puede verse la interfaz de este entorno:

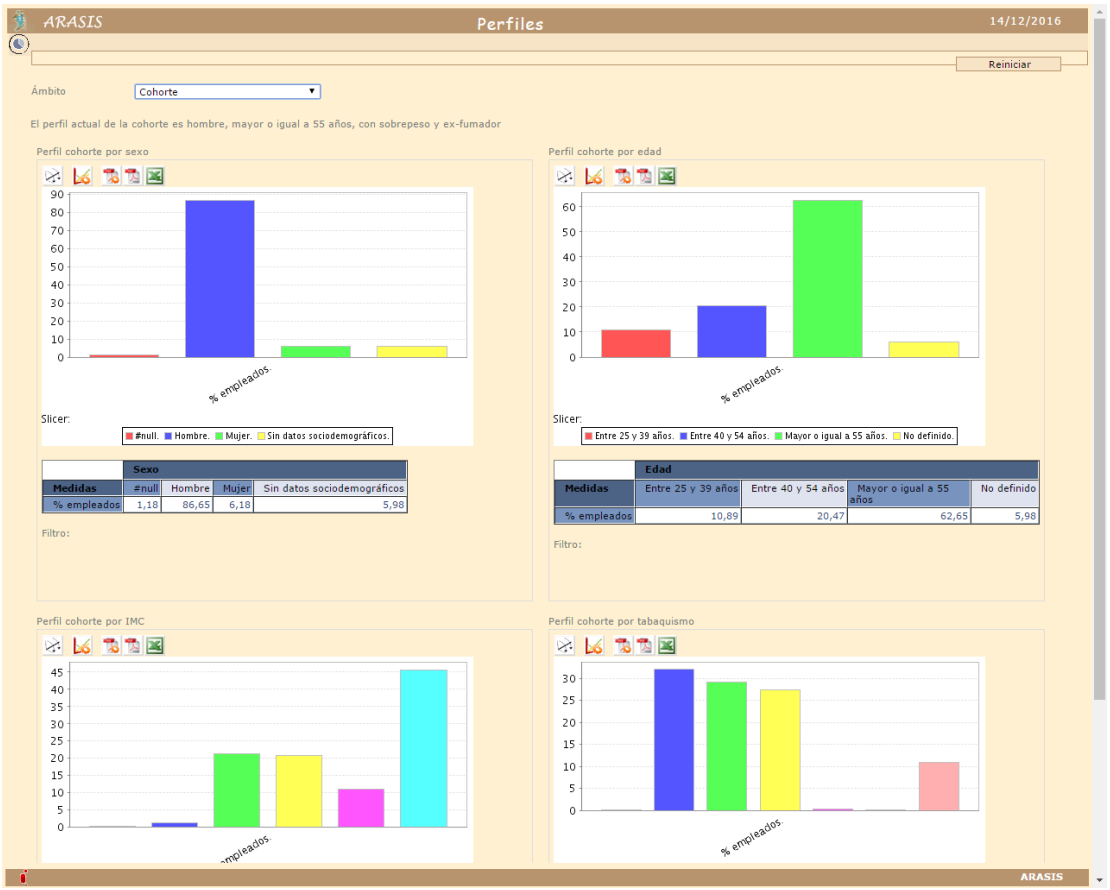


Figura A-12 Interfaz perfiles Sistema BI en ARASIS

En la interfaz podemos ver las dos componentes de la capa de presentación:

- BI Query Manager, en la zona superior. En este caso, la selección únicamente corresponde al tipo de perfil que se quiere obtener, si el perfil del biobanco o de la cohorte.

  En el caso de los perfiles se construyen varias consultas MDX –en el controlador de gestión del entorno- que iniciarán la interfaz en el BI Graphical Manager, ya que se muestran los resultados de varios cubos OLAP a la vez.
- BI Graphical Manager, en la parte inferior. El perfil se muestra como una cadena textual que resume el perfil, construida en la lógica de negocio del BI Retrieval Manager, y tantos resultados de cubos OLAP, mediante la interfaz que proporciona la librería JPivot, como características diferentes definen el perfil.

  Para cada característica se muestra la tabla de resultados del cubo OLAP y un gráfico que los representa. Las funcionalidades de la librería se restringen a la modificación de cada gráfico para personalizarlo y a la exportación de los resultados de cada característica en formato PDF y Excel.

A.8 Desarrollo de cubos OLAP en el sistema BI

A continuación se describe el proceso para construir los cubos OLAP que forman parte del sistema BI y que explotan el conocimiento de la OcBase para los entornos descritos anteriormente.

Como se ha comentado al inicio del apartado, en el sistema BI de ARASIS se han construido 24 cubos OLAP diferentes para satisfacer todas las necesidades del estudio AWHS. Para conocer cómo se construyen, utilizaremos como ejemplo los cubos OLAP que nos permiten analizar la información sobre las ejecuciones de protocolo realizadas en el sistema y la información acerca de los estados de las muestras recogidas en el biobanco.

El proceso de construcción de cada cubo OLAP tiene dos partes:

- Análisis de las dimensiones y métricas que definen el cubo OLAP.
- Construcción del esquema XML que especifica el cubo OLAP, mediante el formato establecido por la librería Mondrian, y de las vistas SQL necesarias para obtener los datos de las dimensiones y la tabla de hechos en el esquema multidimensional (componentes del modelo del sistema BI).

Veamos entonces cómo se construyen los cubos OLAP, empezando por el cubo diseñado para analizar la información de la OcBase acerca de las ejecuciones de protocolos en ARASIS.

En primer lugar, se diseña el esquema multidimensional que define al cubo OLAP, siendo como sigue:

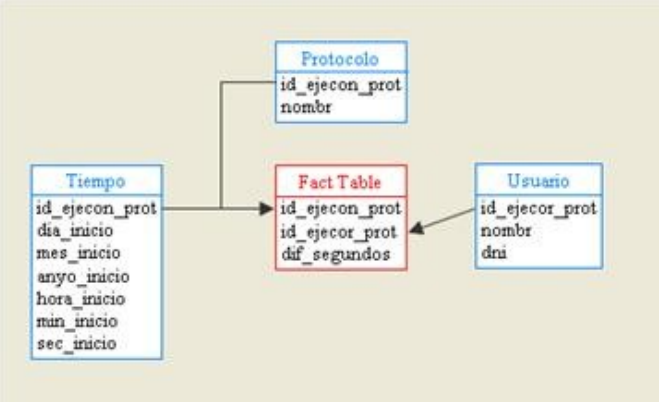


Figura A-13 Esquema cubo OLAP Protocolos Sistema BI en ARASIS

En el esquema OLAP anterior se establecen los siguientes elementos:

- Tabla de hechos: Vista en el que se encuentran las ejecuciones de protocolos de aquellos protocolos que son objeto de estudio, con las referencias de identificador de ejecución de protocolo, de usuario y un campo 'dif_segundos' que contiene la diferencia en segundos en un tipo de dato de doble precisión entre las fechas de fin de ejecución y de inicio del protocolo. Este campo se incluye para los cálculos de tiempo propuestos en las métricas.
- Dimensiones:
  - Protocolo: Dimensión que contiene el nombre del protocolo al que se refiere la ejecución de identificador 'id_ejecon_prot'
  - Usuario. Dimensión que contiene los datos de los usuarios que realizan las ejecuciones de protocolos. Contiene dos datos, que serán especificados como dos jerarquías en la dimensión para poder obtener las métricas en base a los dos:
    - Nombre.
    - DNI.
  - Tiempo. Dimensión que contiene las fechas de ejecución de protocolo como campos de año, mes, día, hora, minuto y segundo. Estos datos se organizarán en niveles de una jerarquía.
- Métricas. Sobre las dimensiones y tabla de hechos anteriores, se definen las siguientes métricas que permitirán realizar los cálculos necesarios sobre las ejecuciones de protocolos:
  - N. º de ejecuciones de protocolos. Obtiene el número de ejecuciones según las dimesiones en una consulta MDX
  - N. º incidentes. Obtiene el número de incidentes en las ejecuciones de protocolos.
  - Total segundos ejecución. Recoge la suma del campo 'dif_segundos' de la tabla de hechos.
  - Total minutos ejecución. El valor de la métrica 'Total segundos ejecución' en valor de minutos.
  - Total horas ejecución. El valor de la métrica 'Total segundos ejecución' en valor de horas.

- % ejecuciones. Obtiene el porcentaje de ejecuciones de un protocolo sobre el total de ejecuciones.
- % incidentes. Obtiene el porcentaje de incidentes que tiene un protocolo sobre el total de incidentes que se han producido.
- % tiempo de ejecución. Obtiene el porcentaje de segundos de ejecución de un protocolo del total de segundos de ejecución.
- N. º medio de incidentes / ejecuciones protocolo. Obtiene la media de incidentes que se producen en las ejecuciones de protocolos
- Media horas ejecución. Obtiene el tiempo medio de ejecución en horas.
- Media minutos ejecución. Obtiene el tiempo medio de ejecución en minutos.
- Media segundos ejecución. Obtiene el tiempo medio de ejecución en segundos.

La tabla de hechos y las dimensiones especificadas en el esquema multidimensional, se construyen como vistas SQL en las componentes del modelo 'OcBase View Manager' y 'Multidimensional Data Base View'.

Una vez especificado el cubo OLAP, el siguiente paso es construirlo en formato XML, siguiendo el esquema y formato establecidos en la librería Mondrian, para que pueda ser reconocido y utilizado tanto por dicha librería como por la librería JPivot, la cual utiliza el mismo formato. Las características de este formato se explican con más detalle en el apartado C.1.2 del Apéndice C.

```
<Schema name="DW_ARASIS">
<Dimension name="Protocolo" caption="Protocolo">
      <Hierarchy hasAll="true" name="Protocolo" allMemberName="Todos" primaryKey="id_ejecon_prot"
caption="Protocolo">
         <Table name="dimprotocolo"/>
         <Level name="Nombre" column="nombr" nameColumn="nombr" ordinalColumn="num"/>
      </Hierarchy>
</Dimension>
<Dimension name="Usuario" caption="Usuario">
      <Hierarchy hasAll="true" name="Nombre" allMemberName="Todos" primaryKey="id_ejecor_prot"
caption="Usuario">
         <Table name="dimusuario"/>
         <Level name="Nombre" column="nombr" nameColumn="nombr"/>
      </Hierarchy>
      <Hierarchy hasAll="true" name="DNI" allMemberName="Todos" primaryKey="id_ejecor_prot" caption="DNI
Usuario">
         <Table name="dimusuario"/>
         <Level name="DNI" column="dni" nameColumn="dni"/>
      </Hierarchy>
</Dimension>
   <Dimension name="Fecha" caption="Fecha">
       <Hierarchy hasAll="true" name="Inicio" allMemberName="Todas" primaryKey="id_ejecon_prot"
caption="Fecha inicio ejecución">
         <Table name="dimtiempoprotocolo"/>
         <Level name="Year" column="anyo_inicio" nameColumn="anyo_inicio"/>
         <Level name="Month" column="mes_inicio" ordinalColumn="mes_num_ini" nameColumn="mes_inicio"
caption="Mes"/>
         <Level name="Day" column="dia_inicio" nameColumn="dia_inicio" caption="Día"/>
         <Level name="Hora" column="hora_inicio" nameColumn="hora_inicio" caption="Hora"/>
         <Level name="Minutos" column="min_inicio" nameColumn="min_inicio" caption="Minuto"/>
         <Level name="Segundos" column="sec_inicio" nameColumn="sec_inicio" caption="Segundo"/>
      </Hierarchy>
</Dimension>

 <Cube name="Cubo_Protocolos_General">
      <Table name="hechosprotocolo"/>
      <DimensionUsage name="Protocolo" source="Protocolo" foreignKey="id_ejecon_prot"/>
      <DimensionUsage name="Usuario" source="Usuario" foreignKey="id_ejecor_prot"/>
      <DimensionUsage name="Fecha" source="Fecha" foreignKey="id_ejecon_prot"/>
      <Measure name="N.º ejecuciones" column="id_ejecon_prot" aggregator="distinct-count"
formatString="######"/>
      <Measure name="Total segundos" column="dif_segundos" visible="false" aggregator="sum"
formatString="#####"/>

      <Measure  name="N.º incidentes" aggregator="sum" formatString="####">
         <MeasureExpression>
            <SQL dialect="generic">
               (select count(distinct ip.id_incid) from incidentesprotocolo ip where hechosprotocolo.id_ejecon_prot =
ip.id_ejecon_prot and incidente='Con incidente')
            </SQL>
         </MeasureExpression>
      </Measure>
```

```
        <CalculatedMember name="Total segundos de ejecución" dimension="Measures" visible="true"
formula="[Measures].[Total segundos]">
            <CalculatedMemberProperty name="FORMAT_STRING" value="#####"/>
        </CalculatedMember>

        <CalculatedMember name="Total minutos de ejecución" dimension="Measures" visible="true"
formula="[Measures].[Total segundos de ejecución]/60">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

        <CalculatedMember name="Total horas de ejecución" dimension="Measures" visible="true"
formula="[Measures].[Total segundos de ejecución]/3600">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

        <CalculatedMember name="% ejecuciones" dimension="Measures" visible="true"
                    formula="([Measures].[N.º ejecuciones] / ([Measures].[N.º
ejecuciones],[Protocolo].DefaultMember,[Usuario.Nombre].DefaultMember,[Fecha].DefaultMember))*100">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

        <CalculatedMember name="% incidentes" dimension="Measures" visible="true"
                    formula="([Measures].[N.º incidentes] / ([Measures].[N.º
incidentes],[Protocolo].DefaultMember,[Usuario.Nombre].DefaultMember,[Fecha].DefaultMember))*100">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

        <CalculatedMember name="% tiempo ejecución" dimension="Measures" visible="true"
                    formula="([Measures].[Total segundos de ejecución] / ([Measures].[Total segundos de
ejecución],[Protocolo].DefaultMember,[Usuario.Nombre].DefaultMember,[Fecha].DefaultMember))*100">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

        <CalculatedMember name="N.º medio incidentes / ejecuciones protocolo" dimension="Measures"
visible="true" formula="[Measures].[N.º incidentes]/[Measures].[N.º ejecuciones]">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>
        <CalculatedMember name="Media segundos por ejecución" dimension="Measures" visible="true"
formula="[Measures].[Total segundos de ejecución]/[Measures].[N.º ejecuciones]">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>
        <CalculatedMember name="Media minutos por ejecución" dimension="Measures" visible="true"
formula="[Measures].[Total minutos de ejecución]/[Measures].[N.º ejecuciones]">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>
        <CalculatedMember name="Media horas por ejecución" dimension="Measures" visible="true"
formula="[Measures].[Total horas de ejecución]/[Measures].[N.º ejecuciones]">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>
    </Cube>
```

Figura A-14 Código cubo OLAP Protocolos Sistema BI en ARASIS

Del código anterior se pueden destacar los siguientes elementos:

- Las tablas que se especifican en las dimensiones y la tabla de hechos, como se ha comentado, son vistas SQL construidas para este esquema multidimensional en las componentes del modelo del sistema BI.
- Las dimensiones se especifican como dimensiones externas al cubo, y dentro de la estructura del cubo OLAP se establece su uso indicando qué clave de la tabla de hechos se utiliza para enlazar la dimensión al cubo OLAP, mediante el atributo foreignKey, así como el nombre con el que se mostrará la dimensión en la tabla de resultados de la interfaz de la librería JPivot.
- En las métricas existen dos tipos:
  - Métricas mediante los elementos 'measure'. Son aquellas métricas que pueden obtenerse directamente de la tabla de hechos por funciones de agregación (count, sum, avg…) o mediante la aplicación de dichas funciones a consultas SQL directas, como la métrica del número de incidentes.
  - Métricas mediante los miembros calculados 'CalculatedMembers'. Los miembros calculados es una utilidad de los cubos OLAP para añadir un miembro de manera manual a una dimensión. En este caso, se utilizan para añadir las métricas a la dimensión de métricas (dimensión interna de los cubos OLAP con todas las métricas definidas en el cubo) para poder realizar cálculos y aplicar funciones MDX sobre las métricas base definidas en los elementos 'measure'.

    Es el caso por ejemplo de la métrica del total de minutos, que realiza un cálculo sobre la métrica de total de segundos (calculada sobre el campo dif_segundos de la tabla de hechos), o las métricas de medias.

  Tanto en las métricas base como en las calculadas, se puede establecer el formato de representación que tendrán sus valores en la tabla de resultados a través de la interfaz de la librería JPivot, así como el nombre con el que se mostrarán en la tabla.

Veamos a continuación otro ejemplo, en este caso, el cubo OLAP diseñado para obtener cálculos respecto a las muestras y sus estados. El esquema multidimensional de este cubo puede verse en la siguiente imagen:

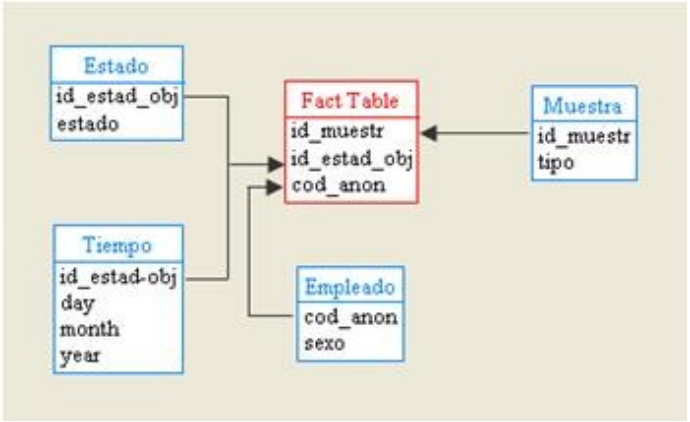


Figura A-15 Esquema cubo OLAP Muestras Sistema BI en ARASIS

En este esquema OLAP, se establecen los siguientes elementos:

- Tabla de hechos: Vista que contiene para cada muestra de la base de datos 'id_muestr', los identificadores del estado de la muestra 'id_estad_obj' y del empleado 'cod_anon' al que pertenece.
- Dimensiones:
  - Muestra. Dimensión que contiene el tipo de muestra –sandre, dna, orina…- al que está referido el identificador 'id_muestr'.
  - Estado. Dimensión que contiene los nombres de los estado actual de la muestra ('Disponible', 'Destruida', 'Alicuotada' y 'Almacenada').
  - Empleado. Dimensión que contiene la información del sexo del empleado. Esta dimensión se incluye para poder obtener información acerca del número de muestras en función del sexo del empleado al que pertenecen.
  - Tiempo. Dimensión que contiene la fecha válida a la que pertenece el estado actual de la muestra. Contiene tres niveles en la jerarquía:
    - Year: Año de la fecha.
    - Month: Mes de la fecha.
    - Day: Día de la fecha.
- Métricas:
  - N. º de muestras. Obtiene el número de muestras distintas según las dimensiones.
  - N. º de empleados. Obtiene el número de empleados distintos que tienen muestras según las dimensiones descritas.
  - % de muestras. Obtiene el porcentaje de muestras sobre las dimensiones descritas.
  - N. º medio de muestras por empleado. Calcula la media de muestras por empleado.

Como en el caso del cubo de protocolos, la tabla de hechos y las dimensiones especificadas en el esquema multidimensional se construyen como vistas SQL en las componentes 'OcBase View Manager' y 'Multidimensional Data Base View'.

Por último, veamos el esquema del cubo en formato XML de la librería Mondrian que especifica el cubo OLAP de muestras, en el que se pueden ver los mismos elementos a destacar que en el cubo de protocolos en cuanto a forma de definición de las dimensiones, métricas y miembros calculados:

```
<?xml version="1.0" encoding="UTF-8"?>
<Schema name="DW_ARASIS">

   <Dimension name="Sexo" caption="Sexo">
      <Hierarchy hasAll="true" name="Sexo" allMemberName="Ambos" primaryKey="cod_anon" caption="Sexo">
         <Table name="dimsociodemograficosultimo"/>
         <Level name="Sexo" column="sexo" type="String"/>
      </Hierarchy>
   </Dimension>

   <Dimension name="Fecha" type="TimeDimension" caption="Fecha inicio estado">
      <Hierarchy hasAll="true" name="Inicio" allMemberName="Todas" primaryKey="id_estad_obj" caption="Fecha
inicio estado">
         <Table name="dimmuestratiempo"/>
         <Level name="Year" column="year" type="String" levelType="TimeYears"/>
         <Level name="Month" column="month" type="String" ordinalColumn="mes" caption="Mes"
levelType="TimeMonths"/>
         <Level name="Day" column="day" type="String" caption="Día" levelType="TimeDays"/>
      </Hierarchy>
   </Dimension>

    <Dimension name="Estado" caption="Estado muestra">
       <Hierarchy hasAll="true" name="Estado" allMemberName="Todos" primaryKey="id_estad_obj"
caption="Estado muestra">
         <Table name="dimmuestraestado"/>
         <Level name="Estado" column="estado" nameColumn="estado"/>
      </Hierarchy>
   </Dimension>

   <Dimension name="Tipo"  caption="Tipo muestra">
      <Hierarchy hasAll="true" name="Tipo" allMemberName="Todos" primaryKey="id_muestr" caption="Tipo
muestra">
         <Table name="tipo_muestra"/>
         <Level name="Tipo" column="tipo" nameColumn="tipo"/>
      </Hierarchy>
   </Dimension>

<Cube name="Cubo_Muestras">

      <Table name="hechosmuestrasactual"/>

      <DimensionUsage name="Tipo" source="Tipo" foreignKey="id_muestr"/>
      <DimensionUsage name="Estado" source="Estado" foreignKey="id_estad_obj"/>
      <DimensionUsage name="Fecha" source="Fecha" foreignKey="id_estad_obj"/>
      <DimensionUsage name="Sexo" source="Sexo" foreignKey="cod_anon"/>
```

```
        <Measure name="N.º muestras" column="id_muestr" aggregator="distinct-count" formatString="######"/>

        <Measure name="Total Empleados" column="cod_anon" visible="false" aggregator="distinct-count"
formatString="######"/>

        <CalculatedMember name="N.º empleados" dimension="Measures" visible="true"
                    formula="CoalesceEmpty ([Measures].[Total Empleados],0)">
            <CalculatedMemberProperty name="FORMAT_STRING" value="#######"/>
        </CalculatedMember>

        <CalculatedMember name="% muestras" dimension="Measures" visible="true"
                    formula="([Measures].[N.º muestras] / ([Measures].[N.º muestras],[Estado].DefaultMember,
[Tipo].DefaultMember,[Fecha].DefaultMember,[Sexo].DefaultMember))*100">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

        <CalculatedMember name="N.º medio muestras por empleado" dimension="Measures" visible="true"
formula="IIF([Measures].[Total Empleados]>0,[Measures].[N.º muestras]/[Measures].[Total Empleados],0)">
            <CalculatedMemberProperty name="FORMAT_STRING" value="##0.00"/>
        </CalculatedMember>

    </Cube>
</Schema>
```

Figura A-16 Código cubo OLAP Muestras Sistema BI en ARASIS

Apéndice B. El 'Timeline web widget' para visualizar datos temporales

Como se ha visto en el apartado anterior de aplicación de la estrategia de diseño a ARASIS, para representar las líneas de vida e historias de los objetos en el sistema BI se ha desarrollado un entorno de interfaz de usuario que permite añadir componentes de visualización de forma dinámica e interactuar con ellos donde se representan las líneas de vida.

En la siguiente imagen podemos volver a ver el entorno:

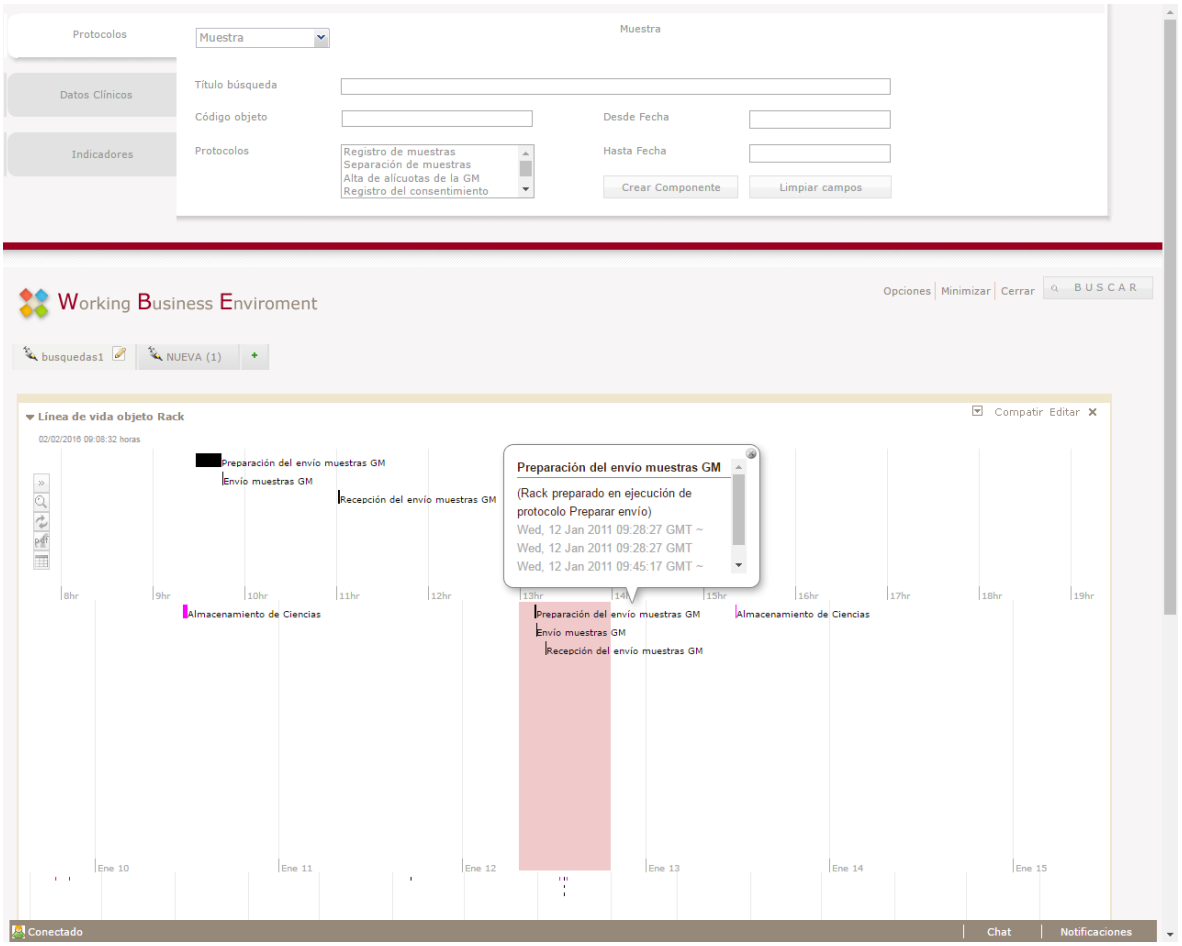


Figura B-1 Librería Simile Timeline en sistema OcTrail de ARASIS

Las componentes que se visualizan en este entorno son líneas de tiempo –*timelines*-, que no son otra cosa que una representación cronológica de un conjunto de eventos. En este caso, el conjunto de eventos son los acontecimientos que constituyen la línea de vida de un objeto.

Para representar las líneas de vida se ha utilizado como base un desarrollo realizado por Simile Projects, concretamente el proyecto Simile Timeline [79].

Este proyecto proporciona una librería Javascript que se puede integrar en aplicaciones propias para generar líneas de tiempo interactivas para el usuario, en las que se pueden incluir un conjunto de eventos sucedidos en un periodo de tiempo fijado para la representación.

Los eventos de la línea de tiempo se incluyen a través de ficheros XML o mediante registros en formato JSON.

En el siguiente extracto de código puede verse un ejemplo de estos eventos en formato XML que se construyen a partir de acontecimientos registrados en la OcBase:

```
<data>
 <event
     start="May 28 2009 09:00:25 GMT"
     end="May 28 2009 10:00:14 GMT "
     isDuration="true"
     title="Alta Alícuotas GM"
     image="http://simile.mit.edu/images/csail-logo.gif">
             Se registran cuatro alícuotas obtenidas a partir de la muestra de un empleado, además de
             relacionarlas con el rack donde serán almacenadas. (Muestra alicuotada en ejecución de
             protocolo Alta de alícuotas de la GM)”
 </event>
 <event
     start="May 29 2009 09:32:01 GMT"
     end="May 29 2009 10:05:15 GMT "
     isDuration="true"
     title="Alta Alícuotas GM"
     image="http://simile.mit.edu/images/csail-logo.gif">
             Se registran cuatro alícuotas obtenidas a partir de la muestra de un empleado, además de
             relacionarlas con el rack donde serán almacenadas. (Muestra alicuotada en ejecución de
             protocolo Alta de alícuotas de la GM)”
 </event>
 <event
     start="Jun 05 2009 16:05:22 GMT"
     end="Jun 05 2009 16:10:37 GMT "
     isDuration="true"
     title="Alta Alícuotas GM"
     image="http://simile.mit.edu/images/csail-logo.gif">
             Se registran cuatro alícuotas obtenidas a partir de la muestra de un empleado, además de
             relacionarlas con el rack donde serán almacenadas. (Muestra alicuotada en ejecución de
             protocolo Alta de alícuotas de la GM)”
 </event>
 <event
     start="Jun 16 2010 12:00:05 GMT"
     title="Almacenamiento de Ciencias"
     >
      Se realiza el almacenamiento de los racks que han sido recibidos. Serán ubicados en una jaula y ésta, a
      su vez, en un congelador.(Rack almacenado en ejecución de protocolo Alta de rack en Ciencias)
 </event>
</data>
```

Figura B-2 Ejemplo de eventos Simile en XML

En el listado anterior, podemos ver el formato que se utiliza en el caso de XML para definir los eventos de la línea de tiempo (en el ejemplo, cuatro acontecimientos de una línea de vida).

Para cada evento, se definen una serie de características que van a permitir representarlo, como, por ejemplo, la fecha de inicio y final en el que se produce el evento, la indicación de si es un evento puntual o por el contrario es un periodo de tiempo –isDuration-, el título o nombre del evento que se mostrará en la línea de tiempo, la imagen que se utilizará en el caso de ser eventos puntuales, etc.

A continuación, se muestra el mismo listado de eventos en formato JSON

```
{
  'wiki-url':"http://simile.mit.edu/shelf/",
  'wiki-section':"Simile JFK Timeline",
  'dateTimeFormat': 'Gregorian',
  'events': [
    {
      'start':" May 28 2009 09:00:25 GMT ",
      'end': "May 28 2009 10:00:14 GMT",
      'title':" Alta Alícuotas GM ",
      'isDuration':true,
      'image': 'http://simile.mit.edu/images/csail-logo.gif',
      'description':" Se registran cuatro alícuotas obtenidas a partir de la muestra de un empleado, además de relacionarlas con el rack donde serán almacenadas. (Muestra alicuotada en ejecución de protocolo Alta de alícuotas de la GM)"
    },
    {
      'start':" May 29 2009 09:32:01 GMT ",
      'end': "May 29 2009 10:05:15 GMT ",
      'title':" Alta Alícuotas GM ",
      'isDuration':true,
      'image': 'http://simile.mit.edu/images/csail-logo.gif',
      'description':" Se registran cuatro alícuotas obtenidas a partir de la muestra de un empleado, además de relacionarlas con el rack donde serán almacenadas. (Muestra alicuotada en ejecución de protocolo Alta de alícuotas de la GM)"
    },
    {
      'start':" Jun 05 2009 16:05:22 GMT ",
      'end': " Jun 05 2009 16:10:37 GMT ",
      'title':" Alta Alícuotas GM ",
      'isDuration':true,
      'image': 'http://simile.mit.edu/images/csail-logo.gif',
      'description':" Se registran cuatro alícuotas obtenidas a partir de la muestra de un empleado, además de relacionarlas con el rack donde serán almacenadas. (Muestra alicuotada en ejecución de protocolo Alta de alícuotas de la GM)"
    },
    {
      'start':" Jun 16 2010 12:00:05 GMT ",
      'title':" Almacenamiento de Ciencias ",
      'description':" Se realiza el almacenamiento de los racks que han sido recibidos. Serán ubicados en una jaula y ésta, a su vez, en un congelador.(Rack almacenado en ejecución de protocolo Alta de rack en Ciencias)
"
    }]
}
```

Figura B-3 Ejemplo de eventos Simile en JSON

Como se puede ver, el contenido de los eventos es el mismo, lo único que cambia es cómo se especifican los eventos en función de un formato u otro.

En el caso de ARASIS, el formato que se ha utilizado es el XML debido a que, como los acontecimientos son obtenidos y preparados para la línea de vida en el negocio del sistema –en el servidor-, se pueden utilizar potentes librerías Java de trabajo con este formato.

Los eventos tienen más características posibles que se pueden utilizar según las necesidades, siendo la fecha inicial y el título las únicas características obligatorias, ya que para el resto de características, en caso de que no se defina, la librería utiliza datos por defecto, como ocurre en la imagen de los eventos, o calculados en función de otros datos, como la fecha fin, que si no se incluye se asume que es igual a la fecha de inicio.

La principal ventaja de esta componente es que permite personalizar su aspecto, así como una amplia variedad de configuraciones en cuanto a número de bandas de tiempo de muestra, formato de los eventos a mostrar, posibilidad de inclusión de información adicional a través de 'burbujas' de información asociadas a un evento…

En el entorno de ARASIS, se desarrolló, además de la personalización de la librería para esta herramienta, los elementos adicionales necesarios para permitir el acceso a las líneas de vida de los objetos relacionados en la historia del objeto. También se añadió una vista tabular de los acontecimientos, la exportación a PDF de los acontecimientos en la línea de vida y la personalización de la información contenida en la burbuja para incluir la información del protocolo ejecutado en el acontecimiento.

Estos elementos añadidos se desarrollaron como una capa por encima de la línea de tiempo, accesibles con los iconos que se localizan a la izquierda de la línea de vida. En la siguiente imagen puede verse la opción de los objetos relacionados desplegada (icono '>>' para mostrarlos e icono '<<' para ocultarlos).

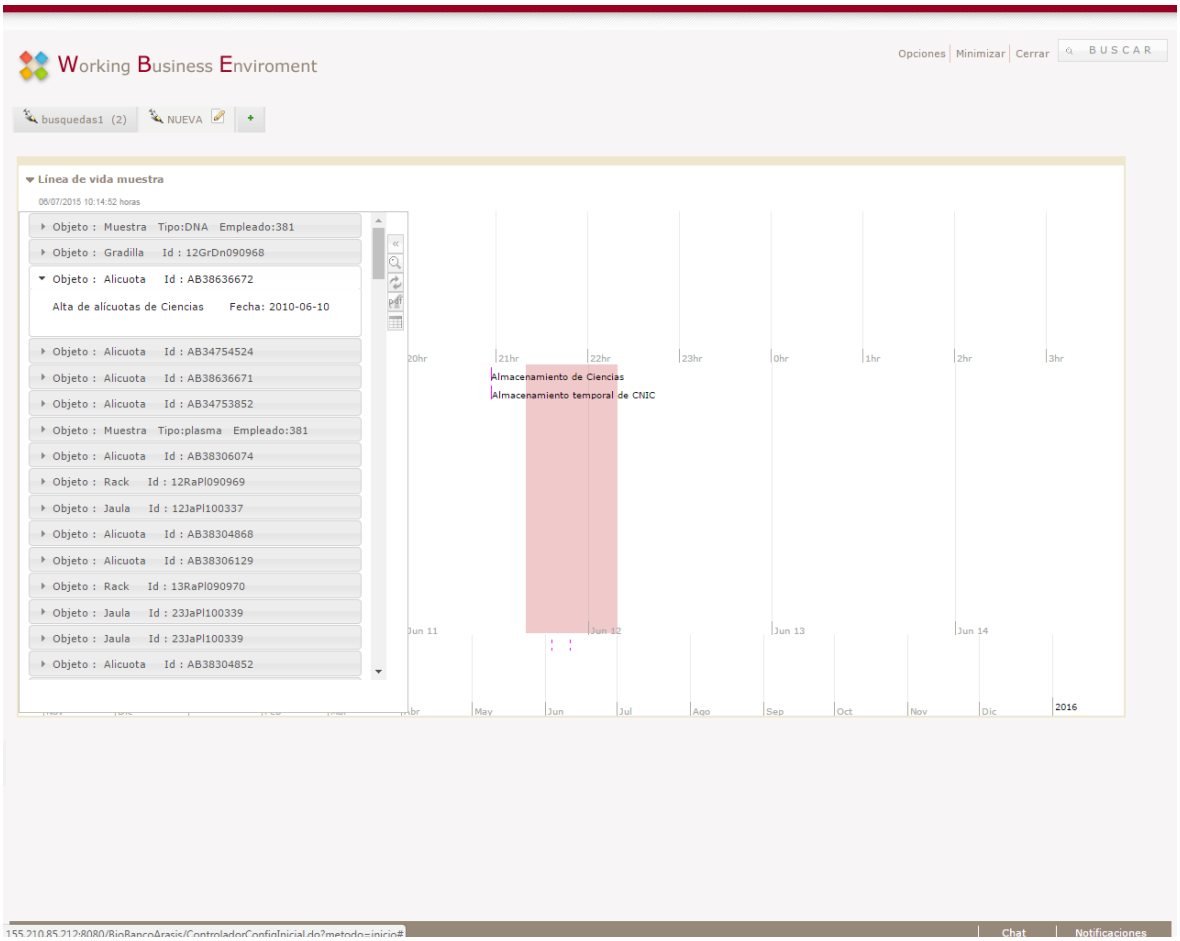


Figura B-4 Elementos adicionales a la librería Simile Timeline construidos en el sistema OcTrail de ARASIS

Apéndice C. Utilización del servidor OLAP Mondrian en ARASIS

Para la implementación y visualización de la tecnología de OLAP y del lenguaje MDX en ARASIS, se utilizan dos librerías Open Source llamadas 'Mondrian' [70] y 'JPivot' [46]. La librería Mondrian es el núcleo de los sistemas OLAP en muchas aplicaciones de BI que proporciona una guía para definir cubos OLAP en formato XML y un motor que permite explotarlos mediante consultas MDX, el lenguaje estándar de consultas de cubos OLAP. La librería 'JPivot' es una librería Open Source que proporciona una interfaz de usuario para la utilización de la tecnología OLAP, utilizando la librería Mondrian como motor OLAP.

Como se ha visto en la aplicación a ARASIS de la estrategia de diseño (en concreto, en la explicación de la arquitectura utilizada), la librería Mondrian se incluye en la componente BI Retrieval Module de la capa de negocio de la arquitectura de un SGBAP. Por su parte, la librería JPivot está en la componente BI Graphical Manager de la capa de presentación y también en la componente BI Retrieval Module, donde configura las consultas MDX para que las ejecute la librería Mondrian.

En los siguientes apartados explicaremos un poco más en detalle cómo funcionan estas librerías y cómo se han utilizado en ARASIS.

C.1 Librería Mondrian

Como se ha comentado, la librería Mondrian proporciona un motor para ejecutar consultas MDX sobre los cubos OLAP definidos en formato XML, utilizando como fuente de datos una base de datos relacional (sistema ROLAP entre los posibles sistemas de funcionamiento OLAP explicados en el apartado 5.3.4).

La librería, construida en lenguaje Java, transforma las consultas MDX, utilizando la especificación del cubo OLAP, en consultas SQL hacia la base de datos que ejecuta mediante conexiones JDBC (interfaz de conexión con bases de datos para aplicaciones desarrolladas en Java).

C.1.1 Arquitectura Mondrian

La arquitectura que utiliza la librería Mondrian puede observarse en la siguiente imagen:

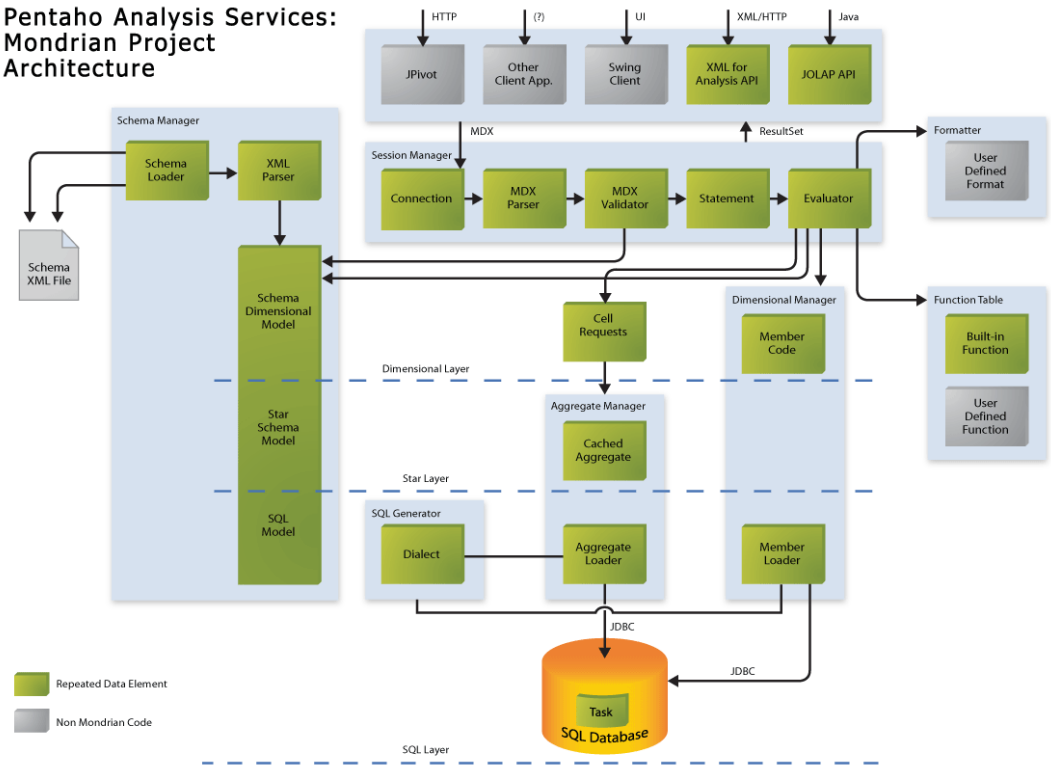


Figura C-1 Esquema de arquitectura librería Mondrian [70]

Como puede verse en la imagen, la arquitectura se divide en cuatro capas: la capa de presentación, la capa 'Dimensional', la capa 'Star' y la capa de almacenamiento o 'SQL layer'. La función de cada capa de la arquitectura es la siguiente.

- Capa de presentación: En esta capa se establece la interfaz del usuario con el sistema OLAP. Para la capa de presentación se pueden utilizar librerías existentes, como la librería JPivot utilizada en ARASIS, o desarrollar interfaces completamente desde cero según las necesidades de cada sistema donde se utilice OLAP, eso sí, desarrolladas en tecnología Java (como Swing o JSP).

  El propósito de estas interfaces es construir las consultas MDX que se requieren ejecutar y traducir los resultados proporcionados por Mondrian en elementos ResultSet (tipo de datos complejo de Java para tratar los datos devueltos por las conexiones JDBC a una base de datos SQL) en elementos gráficos de la interfaz, como tablas dinámicas, gráficos…
- 'Dimensional layer'. Esta capa realiza el análisis, la validación y la ejecución de las consultas MDX. Para ello, traduce las consultas MDX utilizando el esquema XML del cubo OLAP en conjuntos de consultas SQL que llama 'células'.
- 'Star layer'. En esta capa se desarrolla un sistema de caché de las células preparadas en la dimensional layer, devolviendo los resultados si ya dispone de ellos en caché, o enviando las células a la capa SQL para obtener los resultados en caso contrario.
- 'SQL layer'. Se trata del sistema de gestión de bases de datos relacionales propiamente dicho que consulta el motor Mondrian mediante los conjuntos de consultas SQL –células-.

Las capas Dimensional y Star constituyen la librería Mondrian en realidad.

C.1.2 Esquemas XML de cubos OLAP en Mondrian

La librería Mondrian establece para su funcionamiento un esquema XML concreto para construir los cubos OLAP. En ellos, se definen las dimensiones, jerarquías, miembros y cubos que se necesiten desarrollar para el sistema en el que se utilizará OLAP para el análisis de datos. A continuación puede verse un esquema XML simple para un cubo OLAP sobre un caso de ejemplo en el contexto de ARASIS:

```
<Schema>

 <Cube name="cuboDatosClínicos">

    <Table name="tablaHechosDatosClínicos"/>

      <Dimension name="Sexo" foreignKey="id_empleado">
        <Hierarchy hasAll="true" allMemberName="Todos" primaryKey="id_empleado">
          <Table name="datos_empleado"/>
          <Level name="Sexo" column="sexo" uniqueMembers="true"/>
        </Hierarchy>
      </Dimension>

      <Dimension name="Fecha" foreignKey="id_tiempo">
        <Hierarchy hasAll="true" allMemberName="Todos" primaryKey="id_tiempo">
          <Table name="dimFechasAlicuotas"/>
          <Level name="Año" column="anyo" type="Numeric"/>
          <Level name="Mes" column="mes/>
          <Level name="Día" column="dia_mes" type="Numeric"/>
        </Hierarchy>
       </Dimension>

       <Measure name="Número empleados" column="id_empleado" aggregator="distinct-count"
formatString="#,###"/>
       <Measure name=" Altura media" column=" altura" aggregator="avg" formatString="#,###.00"/>
       <Measure name="Peso medio" column="peso" aggregator="avg" formatString="#,###.00"/>
       <CalculatedMember name="Índice masa corporal media" dimension="Measures"
            formula="[Measures].[Peso medio] / ( [Measures].[Altura media] * [Measures].[Altura
media])">
          <CalculatedMemberProperty name="FORMAT_STRING" value="$#,##0.00"/>
       </CalculatedMember>

  </Cube>
</Schema>
```

Figura C-2 Ejemplo sencillo Cubo OLAP

En un primer vistazo, se puede observar que existe un cubo OLAP en el esquema llamado 'cuboDatosClínicos', cuya tabla de hechos es 'tablaHechosDatosClínicos'. Este cubo cuenta con dos dimensiones ('Sexo' y 'Fecha'), y cuatro métricas: tres métricas sencillas (cuenta de número de empleados y medias de alturas y pesos) y una métrica calculada en función de las dos anteriores de medias.

El extracto del cubo OLAP anterior corresponde al cubo construido para los datos longitudinales de CRD registrados en ARASIS.

A continuación se detalla cada elemento principal para explicar su propósito:

- Elemento **'Cube'**:

```
<Cube name="cuboDatosClínicos">
      <Table name="tablaHechosDatosClínicos"/>
      <DimensionUsage name="Sexo" source="Sexo" foreignKey=" id_empleado"/>
      <Measure name="Peso medio" column="peso" aggregator="avg" formatString="#,###.00"/>
</Cube>
```

Figura C-3 Elemento Cube en un Cubo OLAP

El elemento 'Cube' es el elemento principal del esquema XML, y define las características del cubo OLAP. Dentro de un esquema XML pueden definirse varios cubos OLAP mediante este elemento.

Dentro de él, el elemento Table define la tabla de hechos de la que se van a obtener los datos. La tabla de hechos puede ser una tabla o vista existente en la base de datos, o se puede construir como una consulta SQL interna dentro del propio esquema del cubo que obtenga los hechos.

Una vez se ha definido la tabla de hechos, se incluyen las dimensiones que van a configurar el cubo OLAP. Hay dos formas de definir una dimensión:

- Dentro de la estructura del cubo. Esta es la forma en la que se han definido las dimensiones en el ejemplo general del cubo.
- Fuera de la estructura del cubo, referenciándolas en el cubo mediante el elemento DimensionUsage, como en el pequeño ejemplo del elemento 'Cube'. Esta forma se utiliza cuando una misma dimensión se utiliza en varios cubos dentro de un mismo esquema XML.

Mediante el atributo *name* se especifica qué nombre tendrá la dimensión dentro del cubo y con *foreignKey* se indica qué columna de la tabla de hechos es la clave de unión con la dimensión. En el caso de una DimensionUsage, con el atributo *source* se asigna el nombre de la dimensión origen que se ha definido fuera del cubo.

- Elemento '**Dimension'**:

En términos de implementación de la tecnología OLAP a través de Mondrian, una dimensión es una colección de jerarquías y niveles que van a categorizar una métrica por la que se quiere medir una serie de datos. Un ejemplo de dimensión es el siguiente:

```
<Dimension name="Fecha" foreignKey="id_tiempo">
    <Hierarchy hasAll="true" allMemberName="Todos" primaryKey="id_tiempo">
        <Table name="dimFechasAlicuotas"/>
        <Level name="Año" column="anyo" type="Numeric"/>
        <Level name="Mes" column="mes" type="Numeric"/>
        <Level name="Día" column="dia_mes" type="Numeric"/>
   </Hierarchy>
</Dimension>
```

Figura C-4 Elemento Dimension en un Cubo OLAP

En este ejemplo se define una dimensión llamada 'Fecha', que consta de una sola jerarquía de niveles para organizar los datos temporales de la dimensión en tres niveles.

Una dimensión puede tener más de una jerarquía. Por ejemplo, en esta dimensión de tiempo, se podrían tener dos jerarquías, una donde se analizara el tiempo en función de las semanas, y otra en función de los días. Dos jerarquías dentro de una misma dimensión no tienen por qué tener el mismo origen de datos (tabla), sin embargo, sí que tendrán la misma clave de la tabla de hechos a la que referencien para hacer la unión con el cubo (atributo foreign key de la dimensión).

Como se ha comentado, en esta jerarquía hay definidos tres niveles: Año, Mes y Día. Un nivel es una colección de miembros que tienen la misma distancia a la raíz de la jerarquía. En este caso, en el primer nivel estarán los años disponibles, en el segundo, para cada año sus meses, y en el tercero, para cada mes sus días.

La raíz de esta jerarquía está definida por el atributo allMember llamado "Todos", así, todo año será un hijo del miembro "Todos". Si no hubiese allMember (hasAll = false, y sin elemento allMemberName), el elemento raíz o por defecto de la jerarquía sería el primer miembro del nivel Año -2009 por ejemplo-, según la ordenación que se estableciera en ese nivel, pero el resto de miembros del nivel de año no estarían jerarquizados en un nivel inferior a éste.

La necesidad o no de este miembro se deriva del uso que se hará posteriormente de esta jerarquía de la dimensión en las consultas MDX. El miembro defecto 'Todos' es necesario cuando se necesitan realizar cálculos para cualquier miembro que exista en la dimensión como, por ejemplo, saber la media del peso de los empleados independientemente del año.

- Elemento '**Measure'**:

Una vez definidas las dimensiones del cubo, se definen las medidas o métricas. Una medida es un dato calculado a partir de los datos origen de la tabla de hechos. Por ejemplo:

```
<Measure name="Peso medio" column="peso" aggregator="avg" formatString="#,###.00"/>
```

Figura C-5 Elemento Measure en un Cubo OLAP

En este caso se ha definido una medida llamada 'Peso medio', cuyo valor se calculará haciendo la media de la columna 'peso' de la tabla de hechos.

El atributo aggregator define la función con la que se realizará el cálculo, siendo equivalentes a las funciones SQL existentes: sum, count, min, max, avg, distinct-count.

Mediante esta forma se define una medida utilizando la información de la tabla de hechos, sin embargo, Mondrian permite definir medidas utilizando información que no esté directamente en la tabla de hechos, pero que se pueda obtener a través de sus claves, a través de consultas SQL directas:

```
<Measure name="Peso medio" aggregator="avg" formatString="#,###.00">
      <MeasureExpression>
          <SQL dialect="generic">
               (select peso from datos_crd when id_empleado = tablaHechosDatosClínicos.id_empleado)
          </SQL>
      </MeasureExpression>
</Measure>
```

Figura C-6 Medidas por consultas SQL en un Cubo OLAP

En el caso del ejemplo anterior, el peso no lo tendríamos como una columna de la tabla de hechos, y la media la calcularíamos obteniendo su valor directamente de la tabla en la que estuviera registrado, enlazándolo con un identificador de la tabla de hechos (el identificador del empleado en el ejemplo).

Por último, Mondrian también permite definir medidas que resulten de cálculos adicionales de las medidas simples anteriores. Esto se realiza mediante un elemento especial llamado miembro calculado o 'CalculatedMember', que permite añadir un conjunto de miembros directamente a cualquier dimensión utilizando cálculos o funciones del lenguaje MDX, lo que es muy útil para definir subconjuntos de datos o para definir medidas complejas que no podamos definir mediante las funciones de agregación SQL básicas.

```
<Measure name="Peso medio" aggregator="avg" formatString="#,###.00">
      <MeasureExpression>
          <SQL dialect="generic">
               (select peso from datos_crd when id_empleado = tablaHechosDatosClínicos.id_empleado)
          </SQL>
      </MeasureExpression>
</Measure>
```

Figura C-7 Miembros calculados en un Cubo OLAP

Otro ejemplo, imaginemos que tenemos una medida que sume la cantidad de minutos que se tarda en hacer una tarea, llamada 'Total Minutos', y que necesitemos obtener dicho valor también por horas. En este caso, podemos hacer un miembro calculado:

```
<CalculatedMember name="Total Horas" dimension="Measures" visible="true"
                  formula="[Measures].[Total Minutos] / 60">
</CalculatedMember>
```

Figura C-8 Otro ejemplo de miembro calculado en un Cubo OLAP

En estos casos, el cálculo era sencillo, pero dentro del campo fórmula se puede poner cualquier expresión del lenguaje MDX correcta, permitiendo realizar consultas realmente complejas. Por ejemplo, el miembro calculado que obtiene el indicador en los entornos de BI para obtener el número de empleados que tiene alicuotados de cualquier tipo de alícuota:

```
<CalculatedMember name="N.º empleados (alicuotado de todos los tipos)" dimension="Measures"
visible="true"
        formula="CoalesceEmpty(count(distinct(filter([Empleado].[Todos].Children,(([Measures].[N.º alícuotas]
        ,[Tipo].[Todos].[suero])>0) and (([Measures].[N.º alícuotas],[Tipo].[Todos].[metales])>0) and
        (([Measures].[N.º alícuotas],[Tipo].[Todos].[orina])>0) and (([Measures].[N.º alícuotas] ,
        [Tipo].[Todos].[plasma])>0) and (([Measures].[N.º alícuotas],[Tipo].[Todos].[DNA])>0)))),0)">
          <CalculatedMemberProperty name="FORMAT_STRING" value="####"/>
</CalculatedMember>
```

Figura C-9 Ejemplo complejo de miembro calculado en un Cubo OLAP

En el ejemplo anterior, se utilizan las funciones MDX Count, Distinct y Filter, para filtrar miembros de dimensiones según unas determinadas condiciones (que tengan alícuotas de un tipo concreto) y contar los empleados distintos que cumplan esas condiciones, devolviendo el valor resultado ó 0 si no hay ninguno (función CoalesceEmpty).

C.2 Librería JPivot

JPivot es una librería de componentes JSP que permite la visualización de cubos OLAP de forma dinámica.

JPivot se alimenta de otras librerías para su funcionamiento, como es el caso de la librería Mondrian citada anteriormente para la obtención de los datos del sistema OLAP, la librería JFreeChart [45] para la representación de gráficos, así como la librería FOP [83] de java para la exportación de informes en PDF y Excel.

La librería JPivot está licenciada bajo la Common Public License y actualmente la última versión estable publicada es la 1.8.0, aunque los propios desarrolladores de Mondrian actualizan y distribuyen esta librería para mantener la compatibilidad con los cambios que realizan en el motor OLAP, ya que es la librería que utilizan para generar los ejemplos de prueba.

C.2.1 Utilización de la librería JPivot

JPivot se alimenta del cubo OLAP en XML, definido para la librería de Mondrian, para obtener la especificación de las dimensiones y medidas establecidas en el cubo y poder construir así dinámicamente las consultas MDX a realizar sobre él.

La forma de utilizar la librería es mediante dos elementos:

- Declaración del elemento 'jp:mondrianQuery' de la librería JPivot. Este elemento se introduce en una página JSP de la aplicación y permite configurar el enlace que se realiza entre la librería JPivot y la librería Mondrian.

  Una declaración tipo de este elemento puede verse a continuación:

```
<%@ taglib uri="http://www.tonbeller.com/jpivot" prefix="jp" %>
<jp:mondrianQuery id="query01" jdbcDriver="org.postgresql.Driver"
                jdbcUrl="jdbc:postgresql://155.210.85.141:5432/DW_WORKPACKAGES"
                jdbcUser="postgres" jdbcPassword="postgres" catalogUri="cubosOLAP/cuboMuestras.xml">
     SELECT
              {[Measures].[N.º muestras]} ON COLUMNS,
              { [Estado].[Nombre] } ON ROWS
      FROM [Cubo_Muestras]
</jp:mondrianQuery>
```

Figura C-10 Declaración MondrianQuery en JPivot

En este elemento se definen los siguientes atributos:

- Id: indica el nombre que tendrá la consulta MDX y que la identificará en el funcionamiento de la librería JPivot. Así, cuando el usuario interactúe con la librería, toda modificación que implique un cambio en la consulta MDX se aplicará al elemento mondrianQuery localizándolo por este id.
- CatalogUri: establece la ruta donde se encuentra el fichero XML con el esquema que contiene la definición del cubo, tanto para la librería JPivot como para la librería Mondrian.
- Parámetros jdbc: parámetros que utilizará la librería Mondrian para realizar las consultas SQL contra la base de datos según la definición del cubo OLAP.

Como contenido del elemento mondrianQuery se incluye la consulta MDX inicial que se desea realizar al cubo OLAP, y que será modificada posteriormente en la interacción con la librería JPivot.

- Elementos de visualización de la librería JPivot. Una vez se ha definido el elemento de configuración –mondrianQuery-, el siguiente paso es incluir los elementos que van a permitir representar la interfaz de la librería JPivot.

  Los elementos de la librería JPivot se incluyen en un fichero JSP del sistema, y se pueden organizar mediante capas HTML o aplicarle estilos CSS que le modifique el aspecto a la interfaz.

  En el siguiente fragmento se puede ver un ejemplo de código en el que se especifica la interfaz de la librería JPivot utilizada en el sistema BI de ARASIS.

```
<form id="form1" action="olapJSP.jsp" method="post" onsubmit="loa();">

    <div id="contenido_iframeOLAP">

        <%-- include query and title, so this jsp may be used with different queries --%>
        <wcf:include id="include01" httpParam="query" prefix="WEB-INF/queries/" suffix=".jsp"/>

         <%-- define table, navigator and forms --%>
         <jp:table id="table01" query="#{query01}"/>
         <jp:navigator id="navi01" query="#{query01}" visible="false"/>

         <wcf:form id="mdxedit01" xmlUri="/WEB-INF/jpivot/table/mdxedit.xml" model="#{query01}"
visible="false"/>
         <wcf:form id="sortform01" xmlUri="/WEB-INF/jpivot/table/sortform.xml" model="#{query01}"
visible="false"/>

         <jp:print id="print01"/>
         <wcf:form id="printform01" xmlUri="/WEB-INF/jpivot/print/printpropertiesform.xml"
model="#{print01}" visible="false"/>

         <jp:chart id="chart01" query="#{query01}" visible="false"/>
         <wcf:form id="chartform01" xmlUri="/WEB-INF/jpivot/chart/chartpropertiesform.xml"
model="#{chart01}" visible="false"/>
         <wcf:table id="query01.drillthroughtable" visible="false" selmode="none" editable="true"/>
         <jp:selectproperties id="selectprop01" table="#{table01}"  visible="false"/>

         <%-- define a toolbar --%>
         <wcf:toolbar id="toolbar01" bundle="com.tonbeller.jpivot.toolbar.resources">
               <wcf:scriptbutton id="cubeNaviButton" tooltip="toolb.cube" img="cube"
model="#{navi01.visible}"/>
               <wcf:separator/>
                <wcf:scriptbutton id="nonEmpty" tooltip="toolb.non.empty" img="non-empty"
model="#{table01.extensions.nonEmpty.buttonPressed}"/>
                <wcf:scriptbutton id="swapAxes" tooltip="toolb.swap.axes"  img="swap-axes"
model="#{table01.extensions.swapAxes.buttonPressed}"/>
                <wcf:separator/>

                <wcf:scriptbutton id="chartPropertiesButton01" tooltip="toolb.chart.config" img="chart-
config" model="#{chartform01.visible}"/>
                <wcf:scriptbutton id="chartButton01" tooltip="toolb.chart" img="chart"
model="#{chart01.visible}"/>
                <wcf:separator/>
                <wcf:scriptbutton id="printPropertiesButton01" tooltip="toolb.print.config" img="print-
config" model="#{printform01.visible}"/>
                <wcf:imgbutton id="printpdf" tooltip="toolb.print" img="print"
href="./Print?cube=01&type=1"/>
                <wcf:imgbutton id="printxls" tooltip="toolb.excel" img="excel"
href="./Print?cube=01&type=0"/>

           </wcf:toolbar>

            <%-- render toolbar --%>
            <wcf:render ref="toolbar01" xslUri="/WEB-INF/jpivot/toolbar/htoolbar.xsl" xslCache="true"/>

             <%-- render navigator --%>
              <wcf:render ref="navi01" xslUri="/WEB-INF/jpivot/navi/navigator.xsl" xslCache="true"/>
              <wcf:render ref="selectprop01" xslUri="/WEB-INF/jpivot/navi/navigator.xsl"
xslCache="true"/>
```

```
            <%-- edit mdx --%>
            <c:if test="${mdxedit01.visible}">
               <h3>MDX Query Editor</h3>
               <wcf:render ref="mdxedit01" xslUri="/WEB-INF/wcf/wcf.xsl" xslCache="true"/>
            </c:if>

            <%-- sort properties --%>
            <wcf:render ref="sortform01" xslUri="/WEB-INF/wcf/wcf.xsl" xslCache="true"/>

            <%-- chart properties --%>
            <wcf:render ref="chartform01" xslUri="/WEB-INF/wcf/wcf.xsl" xslCache="true"/>

            <%-- print properties --%>
            <wcf:render ref="printform01" xslUri="/WEB-INF/wcf/wcf.xsl" xslCache="true"/>

            <!-- render the table -->
            <p>
               <wcf:render ref="table01" xslUri="/WEB-INF/jpivot/table/mdxtable.xsl" xslCache="true"/>
            </p>
            <p class="tipo_letra">
               Filtro:
               <wcf:render ref="table01" xslUri="/WEB-INF/jpivot/table/mdxslicer.xsl" xslCache="true"/>
            </p>

            <!-- drill through table -->
            <wcf:render ref="query01.drillthroughtable" xslUri="/WEB-INF/wcf/wcf.xsl" xslCache="true"/>

            <!-- render chart -->
            <wcf:render ref="chart01" xslUri="/WEB-INF/jpivot/chart/chart.xsl" xslCache="true"/>
         </div>
</form>
```

Figura C-11 Declaración interfaz JPivot en ARASIS

Como se puede ver en el código, se utilizan tanto elementos propios definidos en la librería JPivot, los que comienzan por "jp:", como elementos de la librería WCF [93], una librería de componentes JSP que utiliza la librería JPivot.

En este código, los elementos más importantes son:

- Elemento form: permite establecer el controlador JPivot que recibirá las peticiones del usuario mediante la librería.
- Elemento "wcf:include id='include01' httpParam='query'". Permite definir en qué archive se encuentra definido el element mondrianQuery, si está definido en un archivo diferente.
- Etiquetas de definición de elementos. Algunos elementos se definen mediante las etiquetas JPivot y otros mediante las etiquetas WCF. Por ejemplo:
  - Dentro de las etiquetas JPivot, destacan el elemento "jp:table", que configura la tabla en la que se mostrarán los resultados del cubo OLAP –notar la referencia al elemento mondrianQuery con el 'query01'-, o la etiqueta "jp:chart", que configura el elemento para representar gráficos.
  - Dentro de las etiquetas de WCF, destaca la etiqueta "wcf:form" que permitirá

mostrar los formularios de configuración de propiedades de gráficos o exportación de PDF, o la etiqueta "wcf:toolbar", con sus etiquetas internas "wcf:scriptbutton", que permiten definir la barra de iconos de la interfaz.

- Etiquetas "wcf:render". A través de estas líneas de código se le indica a la librería JPivot, utilizando los mecanismos que le proporciona la librería WCF que utiliza, dónde tiene que representar cada elemento definido con las etiquetas anteriores.

Como se ha comentado, la librería JPivot se puede personalizar en apariencia y en forma de visualización, incluyendo los códigos anteriores en los ficheros JSP que se necesiten o modificando los ficheros de estilos CSS que tiene la librería.

En la siguiente imagen se puede ver el resultado en ARASIS del código anterior, como vimos en la explicación del sistema BI de la arquitectura.

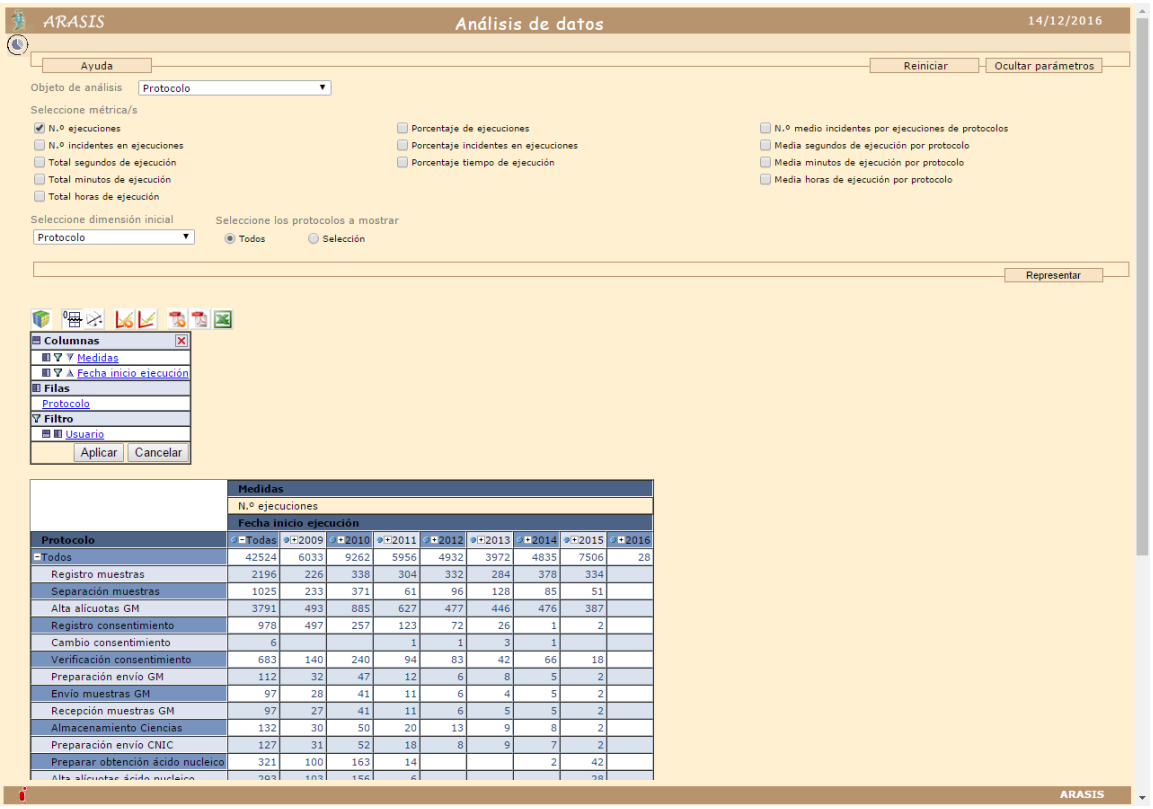


Figura C-12 Resultado interfaz JPivot en ARASIS

En este caso, se muestra en una capa inferior de la pantalla junto al BI Query Manager. Sin embargo, en otros entornos donde se ha incluido el funcionamiento OLAP se ha modificado considerablemente su aspecto:

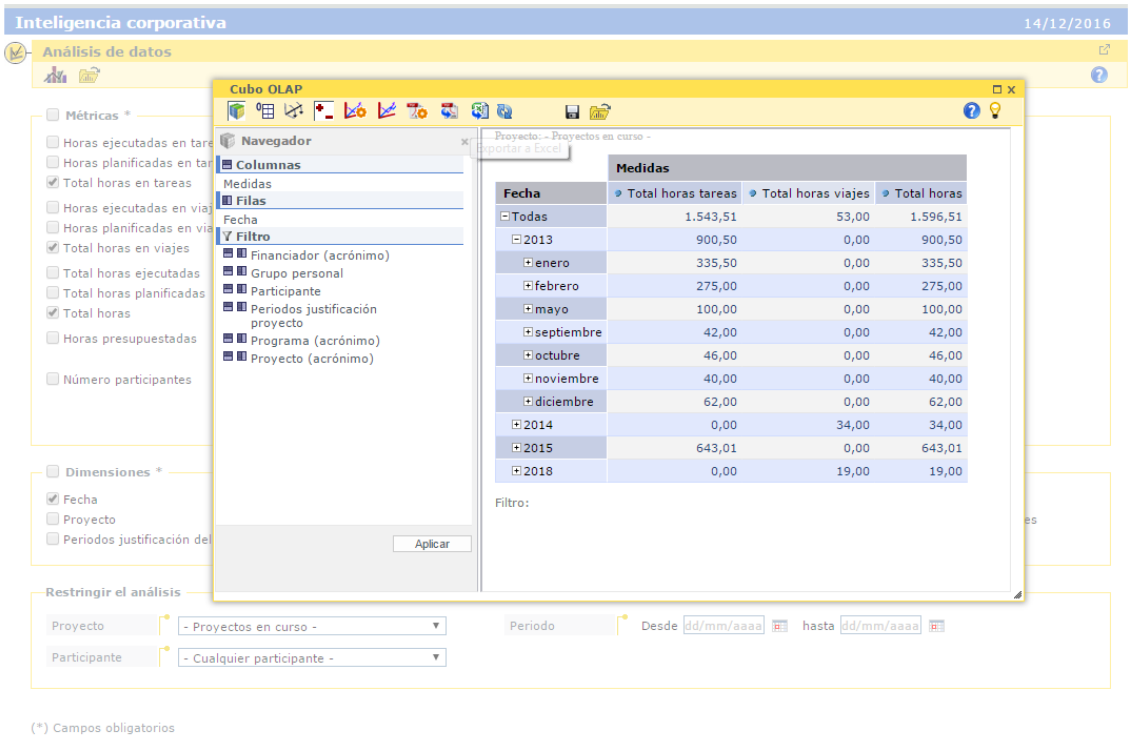


Figura C-13 Otro ejemplo de interfaz JPivot

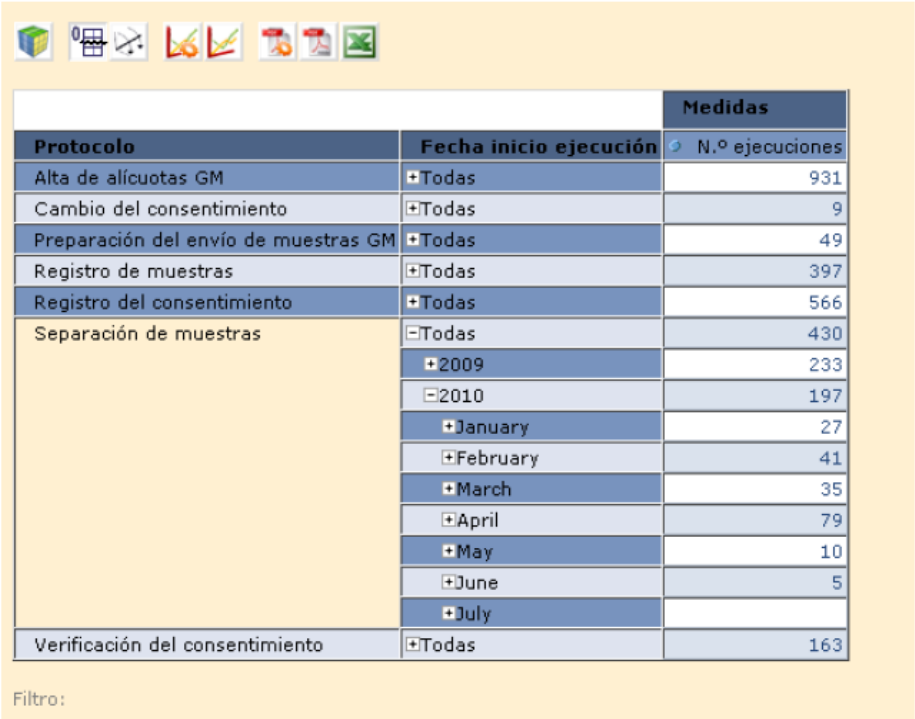
Como se puede ver en la imagen anterior, el entorno de OLAP es muy diferente, con el funcionamiento de la librería JPivot en una ventana emergente por encima del BI Query Manager y unos estilos de visualización completamente modificados.

C.2.2 Funcionalidades para la realización de análisis

La librería JPivot ofrece la posibilidad de utilizar un conjunto de funcionalidades para modificar dinámicamente el análisis que se está realizando sobre el cubo OLAP mediante la propia interfaz de usuario.

En el caso de ARASIS, las funcionalidades que se incluyeron en la interfaz de los entornos son las siguientes:

- Desagrupar/agrupar resultados. Esta función está dentro de los resultados de la tabla, mediante los iconos '+' y '-'.



		Medidas
Protocolo	Fecha inicio ejecución	N.º ejecuciones
Alta de alícuotas GM	+Todas	931
Cambio del consentimiento	+Todas	9
Preparación del envío de muestras GM	+Todas	49
Registro de muestras	+Todas	397
Registro del consentimiento	+Todas	566
Separación de muestras	-Todas	430
	+2009	233
	-2010	197
	+January	27
	+February	41
	+March	35
	+April	79
	+May	10
	+June	5
	+July	
Verificación del consentimiento	+Todas	163

Filtro:

Figura C-14 Desagrupar / agrupar resultados JPivot

La imagen anterior muestra una serie de protocolos, y la dimensión 'Fecha' agrupadas en todos los protocolos menos en uno, en el que se ha desagrupado por años, y a su vez el año 2010 por meses.

- Modificar información seleccionada

  A través del icono en forma de cubo de la barra de iconos, se pueden modificar los datos que se están visualizando en la tabla.

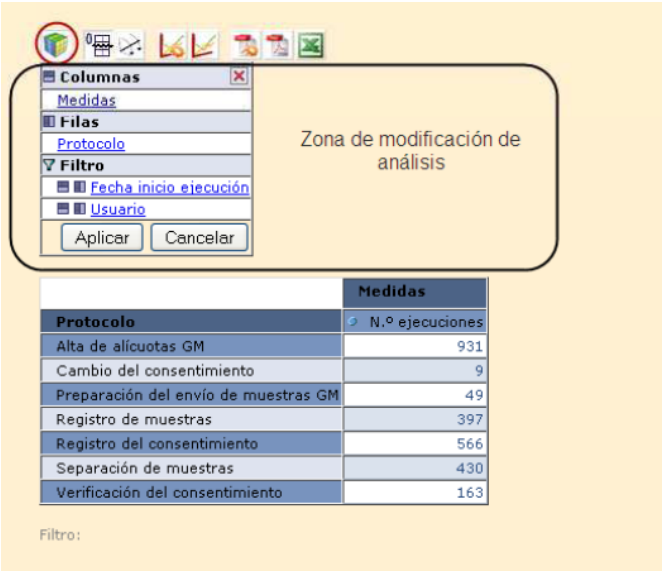


Figura C-15 Modificar análisis JPivot

A través del icono se accede a la modificación del análisis donde la información de la tabla de resultados se organiza en tres zonas:

- Columnas: se muestran las dimensiones que se están visualizando en las columnas de la tabla de datos.
- Filas: se muestran las dimensiones que se están visualizando en las filas de la tabla de datos.
- Filtro: se muestran el resto de dimensiones.

A través de estas zonas se puede realizar las siguientes modificaciones del análisis:

- Añadir, ocultar o mover dimensiones en la tabla. Así, se pueden añadir u ocultar dimensiones que se visualizan en la tabla utilizando los iconos que se muestran al lado de los nombres de las dimensiones. Ocultar una dimensión es colocarla en la zona de filtros.

  En la siguiente imagen puede verse cómo se añade una columna a la tabla de resultados.

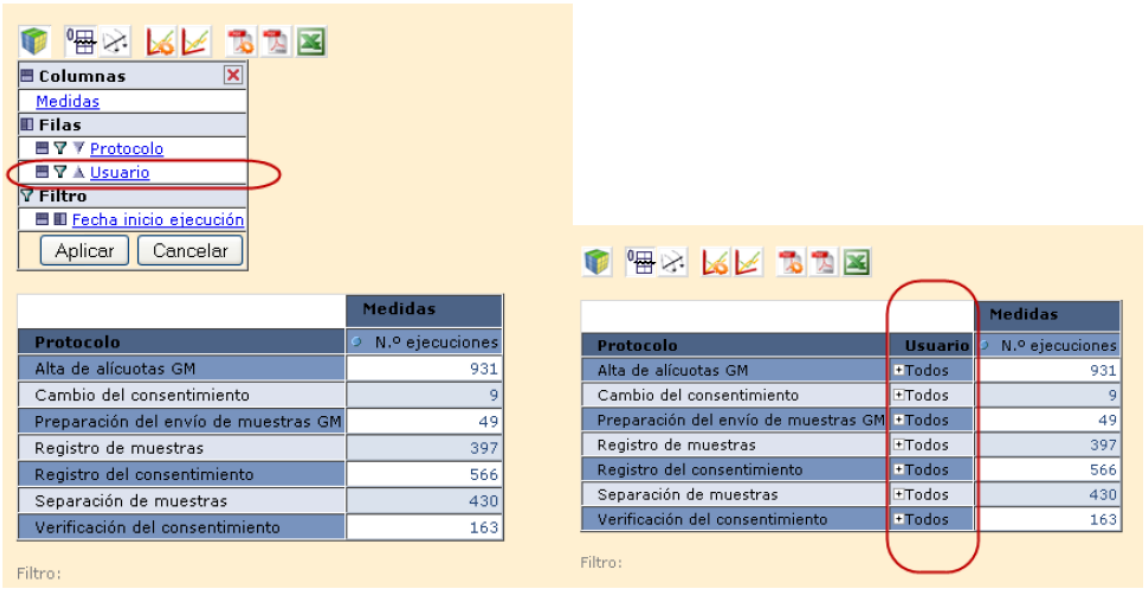


Figura C-16 Añadir dimensión JPivot

- Organizar el orden de las filas o columnas. Cuando en las columnas o las filas de las tablas se muestran varias dimensiones, como en el ejemplo de la imagen anterior, se puede variar el orden en el que se muestran en la tabla de resultados mediantes los iconos en forma de flecha al lado de los nombres.
- Ocultar o añadir datos de una dimensión. Además de la funcionalidad de añadir u ocultar dimensiones, se puede realizar esta acción también sobre los datos de una dimensión concreta. Si se pulsa sobre el nombre de la dimensión en esta zona de modificación, se accede a los miembros disponibles de ella donde se pueden seleccionar o deseleccionar los miembros que se quieran mostrar.

  En la siguiente imagen puede verse un ejemplo, en el que se están mostrando todos los meses de un año y se selecciona para sólo dejar el mes de marzo:

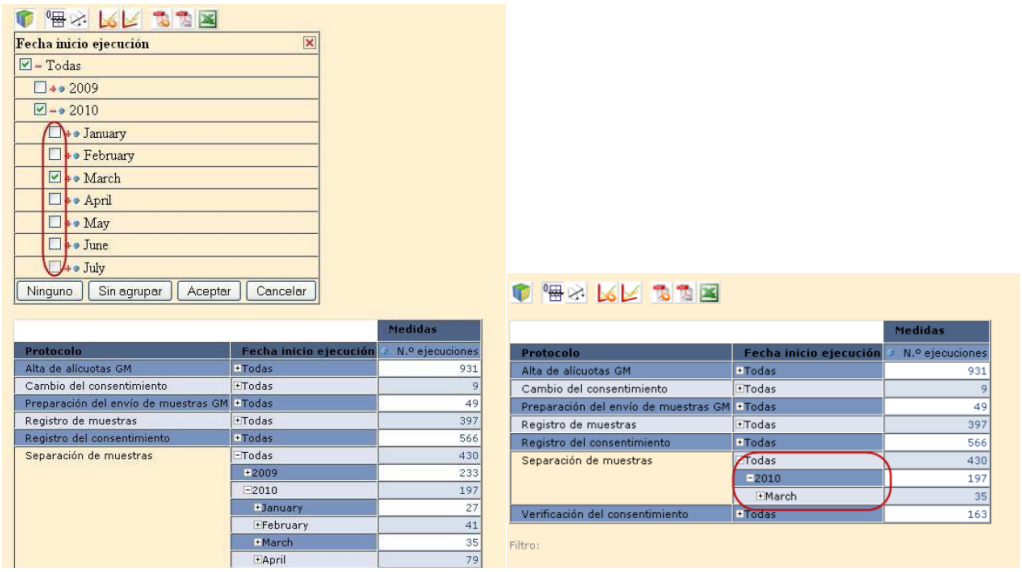


Figura C-17 Seleccionar datos JPivot

- Añadir condición de filtro. De la misma forma que se añaden u ocultan datos de una dimensión, se pueden añadir condiciones de filtro para que todo valor de la tabla de resultados cumpla esas condiciones. Esto se realiza pulsando, en vez de sobre las dimensiones de columnas o filas, en las dimensiones que estén en la zona de filtro.

  En la siguiente imagen puede verse un ejemplo. En él, en vez de seleccionar que sólo se muestre el mes de marzo en la dimensión fecha, se ha pasado esta dimensión a la zona de filtro y se ha aplicado como condición que el mes sea marzo. El resultado es que para todos los protocolos el valor mostrado corresponde a las ejecuciones realizadas exclusivamente en marzo.

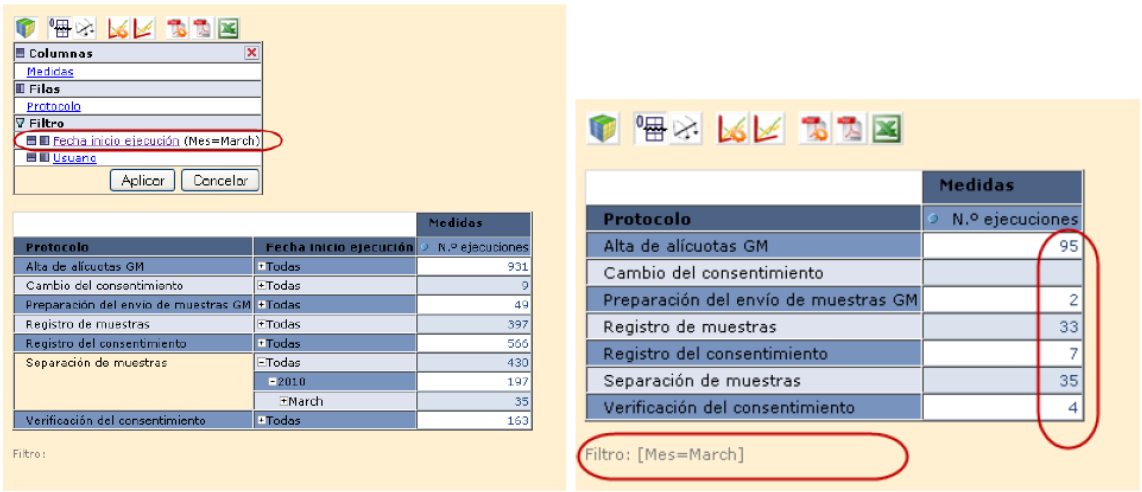
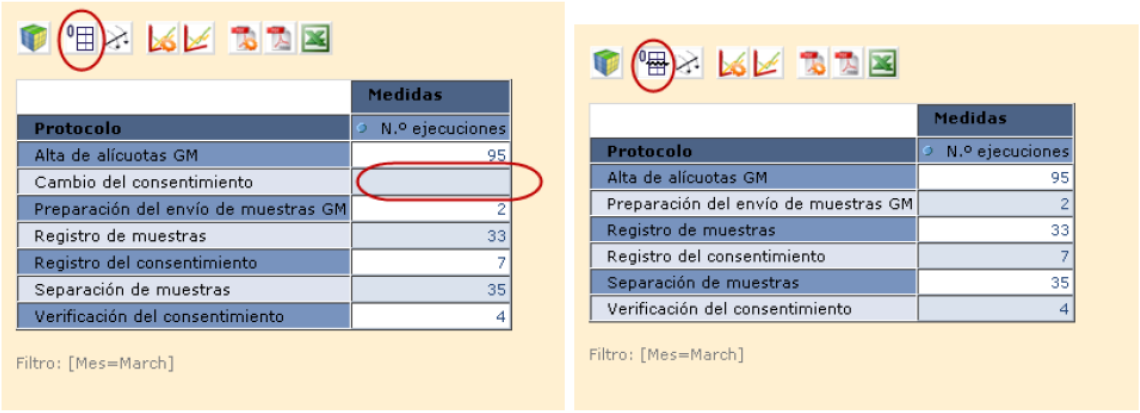


Figura C-18 Filtrar datos análisis JPivot

- Eliminar resultados vacíos. En ocasiones, el cálculo de una métrica puede ocasionar que para determinados miembros de las dimensiones no haya resultados. En estos casos pueden ocultarse dichos datos de la tabla para evitar datos superfluos.



	Medidas
Protocolo	N.º ejecuciones
Alta de alícuotas GM	95
Cambio del consentimiento	
Preparación del envío de muestras GM	2
Registro de muestras	33
Registro del consentimiento	7
Separación de muestras	35
Verificación del consentimiento	4

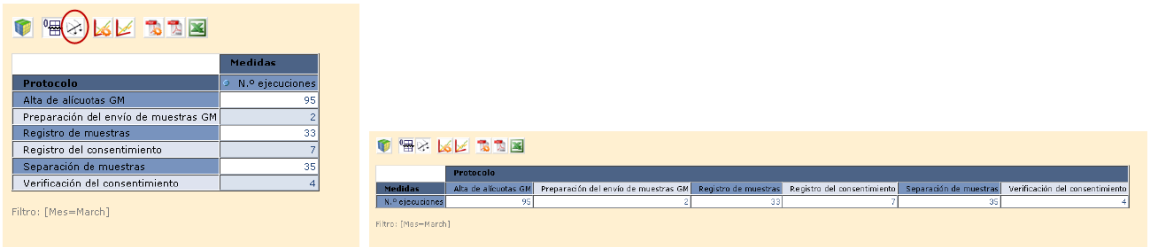
Filtro: [Mes=March]

	Medidas
Protocolo	N.º ejecuciones
Alta de alícuotas GM	95
Preparación del envío de muestras GM	2
Registro de muestras	33
Registro del consentimiento	7
Separación de muestras	35
Verificación del consentimiento	4

Filtro: [Mes=March]

Figura C-19 Eliminar resultados vacíos JPivot

- Intercambio de ejes. Las columnas y filas que se muestran en la tabla de resultados pueden intercambiarse entre sí, pasando las filas a ser columnas y viceversa.



	Medidas
Protocolo	N.º ejecuciones
Alta de alícuotas GM	95
Preparación del envío de muestras GM	2
Registro de muestras	33
Registro del consentimiento	7
Separación de muestras	35
Verificación del consentimiento	4

Filtro: [Mes=March]

Medidas	Protocolo					
	Alta de alícuotas GM	Preparación del envío de muestras GM	Registro de muestras	Registro del consentimiento	Separación de muestras	Verificación del consentimiento
N.º ejecuciones	95	2	33	7	35	4

Filtro: [Mes=March]

Figura C-20 Intercambiar ejes JPivot

- Mostrar/ ocultar gráficos. Asociados a los datos que se muestran en la tabla, es posible la representación de gráficas para visualizarlos en dicho formato.

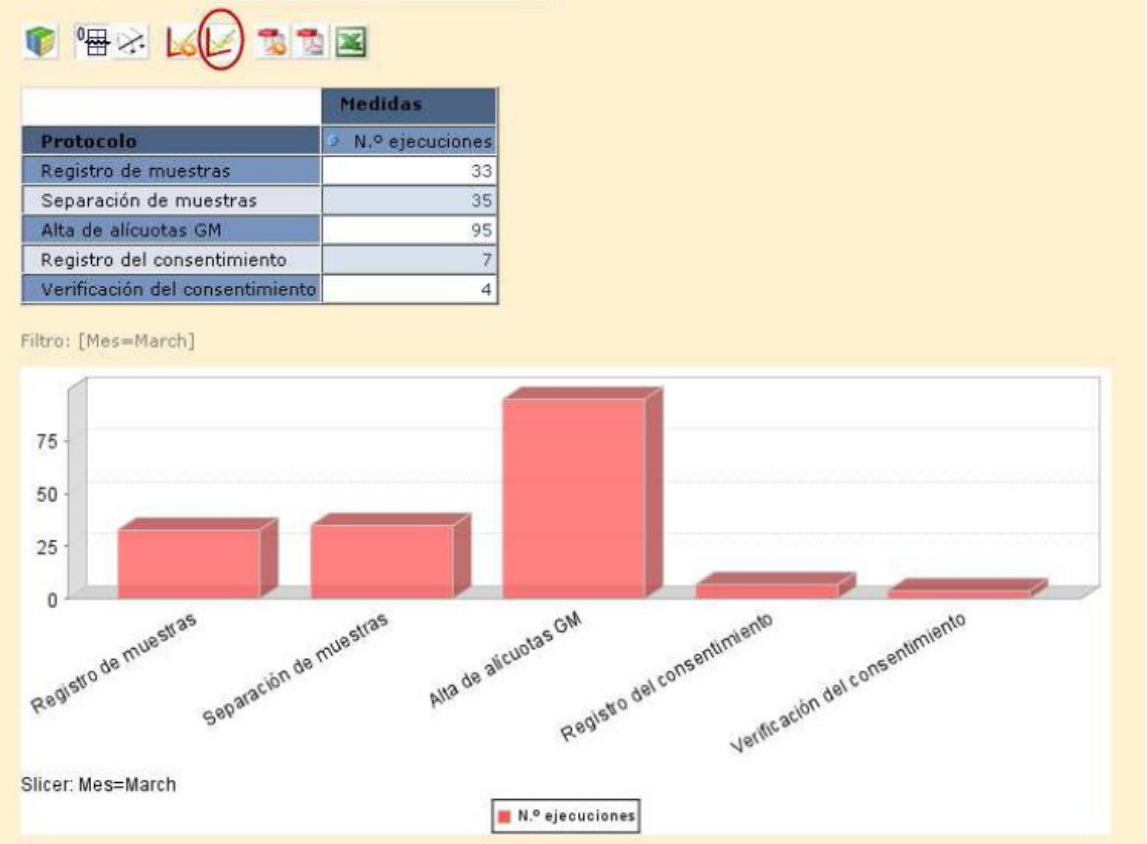


Figura C-21 Gráficos JPivot

Los gráficos pueden personalizarse. Se puede definir desde la forma de la gráfica (en barras, en líneas, de tarta…), hasta la fuente de las etiquetas, título y tamaño de la gráfica.

- Exportación de datos. Los datos pueden exportarse tanto en PDF como en Excel, a través de los iconos correspondientes con forma de archivo.

  En el caso de la exportación del PDF, ésta se puede personalizar para indicar, por ejemplo, el tamaño de la página del fichero (Din A4, Din A3) o la orientación de la página (horizontal o vertical).

Bibliografía

[1] Aldawud O., Elrad T., Bader A., "UML profile for aspect-oriented software development", The Third International Workshop on Aspect Oriented Modeling, 2003.

[2] Alexander C., "A pattern language: towns, buildings, construction", Oxford University Press, 1977.

[3] Allué, A., López, A., Ciria, J. C., Domínguez, E., Francés, Á., Zapata, M. A. (2016, July). The task-oriented occurrence pattern. In Proceedings of the 21st European Conference on Pattern Languages of Programs (p. 6). ACM. http://dx.doi.org/10.1145/3011784.3011790.

[4] Andany J., Léonard M., Palisser C., "Management of schema evolution in databases", in Proc. VLDB, Barcelona, Spain, pp. 161–170, 1991.

[5] Andreasson H. et al., "Autonomous transport vehicles: where we are and what is missing", Robotics & Automation Magazine, IEEE 22.1, pp. 64–75, 2015.

[6] Apache Struts, URL: https://struts.apache.org/. Último acceso: marzo 2017.

[7] Asghar M.R., Ion M., Russello G., Crispo B, "Securing Data Provenance in the Cloud", Open Problems in Network Security, Lecture Notes in Computer Science, vol 7039. Springer, 2012.

[8] ATIS Committee PRQC , "ATIS Telecom Glossary 2012 - audit trail", 2012, URL: http://www.atis.org/glossary/definition.aspx?id=5572. Último acceso: marzo 2017.

[9] Atzori L., Iera A., Morabito G., "The Internet of Things: A survey", Computer Networks, vol. 54 (15), pp. 2787-2805, 2010.

[10] AXELOS, ITIL, IT Infrastructure Library, URL: https://www.axelos.com/best-practice-solutions/itil. Último acceso: marzo 2017.

[11] Bézivin J., "Model driven engineering: An emerging technical space" in Proc. GTTSE, pp. 36–64, Springer Berlin Heidelberg, 2006.

[12] Bonabeau, E.; Corne, D.; Knowles, J.; Poli, R. "Swarm intelligence theory: A snapshot of the state of the art", Theoretical Computer Science, vol. 411, no. 21, pp. 2081-2083, mayo 2010.

[13] Brenzler J., Clark S.J., "Longitudinal Database Design", The INDEPTH Annual General Meeting, 2000.

[14] Campanile F., Coppolino L., Giordano S., Romano L., "A business process monitor for a mobile phone recharging system," J. Syst. Archit., vol. 54, no. 9, pp. 843–848, septiembre 2008.

[15] Casasnovas J. A. et al, "Aragon workers' health study – Design and cohort description", BMC Cardiovasc. Disord., vol. 12, no. 1, p. 45, 2012.

[16] Castro C. D., Grandi F., Scalas M. R., "Schema versioning for multitemporal relational databases", Inform. Syst., vol. 22, no. 5, pp. 249–290, 1997.

[17] Cattrysse D. G., Van Wassenhove L. N., "A survey of algorithms for the generalized assignment problem", European Journal of Operational Research, vol. 60, no. 3, pp. 260-272, ISSN 0377-2217, 1992.

[18] Chaâbane M. A., Andonoff E., Bouzguenda L., Bouaziz R., "Dealing with business process evolution using versions", in Proc. ICE-B, Porto, Portugal, pp. 267–278, 2008.

[19] Chaudhuri S., Dayal U., "An overview of data warehousing and OLAP technology", SIGMOD Rec., vol. 26, no. 1, pp. 65–74, marzo 1997.

[20] Cheney J., Chiticariu L., Tan W.-C., "Provenance in Databases: Why, How, and Where", Foundations and Trends in Databases, vol. 1 no. 4, pp. 379-474, abril 2009.

[21] Clark S.J., "A temporal data model and the structured population event history register", Demographic Research 15, 181-252, 2006.

[22] Clark S.J., "An introduction to the General Temporal Data Model and the Structured Population Event History Register", Scandinav Journal of Public Heath, 21-25, 2007.

[23] Curbera F., Doganata Y., Martens A., Mukhi N.K., Slominski A., "Business provenance–a technology to increase traceability of end-to-end operations", vol 5331. In: OTM Conferences (1). Lecture Notes in Computer Science, vol 5331, pp 100–119, Springer, 2008.

[24] Curino C., Moon H. J., Zaniolo C., "Graceful database schema evolution: The PRISM workbench", Proc. VLDB, vol. 1, no. 1, pp. 761–772, 2008.

[25] DB-MAIN: a data-architecture tool, REVER, URL: http://www.rever.eu/en/content/db-main-homepage. Último acceso: marzo 2017.

[26] Diwas Singh KC, "Does multitasking improve performance? Evidence from the emergency department", Manufacturing & Service Operations Management 16, 2, 168–183, 2013.

[27] Domínguez, E., Pérez, B., Rubio, Á. L., Zapata, M. A., Allué, A., Lavilla, J., "Occurrence-oriented design strategy for developing business process monitoring systems", IEEE Transactions on Knowledge and Data Engineering, 26(7), 1749-1762, 2014.

[28] Domínguez E., Pérez B., Zapata M. A., "Towards a traceable clinical guidelines application: A model driven approach," Methods Inform. Med., vol. 46, no. 6, pp. 571–580, 2010.

[29] Domínguez E., Lloret J., Rubio A. L., Zapata M. A., "Medea: A database evolution architecture with traceability", Data Knowl. Eng., vol. 65, no. 3, pp. 419–441, 2008.

[30] Domínguez E., Lloret J., Pérez B., Rodríguez A., Rubio A. L., Zapata M. A., "Evolution of XML schemas and documents from stereotyped UML class models: A traceable approach", Inform. Softw. Technol., vol. 53, no. 1, pp. 34–50, 2011.

[31] Domínguez E., Pérez B., Rubio A. L., Zapata M. A., Allué A., López A., "Developing provenance-aware query systems: an occurrence-centric approach". Knowledge and Information Systems 50, pp 661–688, 2017.

[32] EuroPloP, European Conference on Pattern Languages of Programs, URL: http://www.europlop.net/content/conference-0. Último acceso: marzo 2017.

[33] Fletcher P., "Infinity", Philosophy of Logic, vol. 5 of the Handbook of the Philosophy of Science, pp.523–585, Elsevier, ISBN 0-444-51541-0, 2007.

[34] Gamma E., Helm R., Johnson R., Vlissides J., "Design Patterns: Elements of Reusable Object-Oriented Software", Reading, MA, USA: Addison Wesley, 1995.

[35] Gangadharan G. R., Swami S. N., "Business intelligence systems: design and implementation strategies," 26th International Conference on Information Technology Interfaces, Cavtat, pp. 139-144 Vol.1, 2004.

[36] Gerede C. E., Bhattacharya K., Su J., "Static analysis of business artifact-centric operational models," in Proc. SOCA, Newport Beach, CA, USA, pp. 133–140, 2007.

[37] Gerkey B. P., Mataric M. J., "A formal analysis and taxonomy of task allocation in multi-robot systems", The International Journal of Robotics Research 23.9, pp. 939–954, 2004.

[38] Gerkey B. P., Mataric M. J., "Sold!: Auction methods for multirobot coordination", Robotics and Automation, IEEE Transactions on 18.5, pp. 758–768, 2002.

[39] Ghattas J., Soffer P., Peleg M., "Improving business process decision making based on past experience", Decision Support Systems, vol. 59, pp. 93-107, ISSN 0167-9236, Marzo 2014.

[40] Golfarelli M., Lechtenbörger J., Rizzi S., Vossen G., "Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation", Data Knowl. Eng., vol. 59, no. 2, pp. 435–459, 2006.

[41] Grandi F., "A relational multi-schema data model and query language for full support of schema versioning", in Proc. 10th SEBD, pp. 323–336, 2002.

[42] Hillier F. S., Lieberman G. J., "Introduction to Operations Research", McGraw-Hill, Boston MA, 10th Edition, 2014.

[43] ISO, ISO 9000:2015 Quality management systems -- Fundamentals and vocabulary. URL: https://www.iso.org/standard/45481.html. Último acceso: marzo 2017.

[44] ISO 9000:2005, Sistemas de gestión de la calidad - Fundamentos y vocabulario. URL: http://www.umc.edu.ve/pdf/calidad/normasISO/Norma_ISO_9000_2005.pdf. Traducción certificada. Último acceso: marzo 2017.

[45] JFreeChart, URL: http://www.jfree.org/jfreechart/. Último acceso: marzo 2017.

[46] JPivot, a JSP based OLAP client, URL: http://jpivot.sourceforge.net/. Último acceso: marzo 2017.

[47] Kang B., Kim D., Kang S.-H., "Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction," Expert Syst. Appl., vol. 39, no. 5, pp. 6061–6068, 2012.

[48] Kang D., Lee S., Kim K., Lee J. Y., "An OWL-based semantic business process monitoring framework," Expert Syst. Appl., vol. 36, no. 4, pp. 7576–7580, 2009.

[49] Knuth, Donald E., "Backus Normal Form vs. Backus Naur Form", Communications of the ACM. 7 (12): 735–736, 1964.

[50] Ko R. K. L., "A computer scientist's introductory guide to business process management (BPM)," Crossroads, vol. 15, no. 4, pp. 4:11–4:18, junio 2009.

[51] Labella T. H., Dorigo M., Deneubourg J.-L., "Selforganised task allocation in a group of robots", Distributed Autonomous Robotic Systems 6, Springer, pp. 389–398, 2007.

[52] LISBioBank. Ministerio de Industria, Turismo y Comercio, TSI-020302-2008-8, Idea Informática S. A. Infozara S. L., Universidad de Zaragoza, 2008-2009.

[53] Lucas P., "Quality checking of medical guidelines through logical abduction", in Proc. AI-2003, pp. 309–321, 2003.

[54] Lucia A. D., Deufemia V., Gravino C., Risi M., "Design pattern recovery through visual language parsing and source code analysis", J. Syst. Softw., vol. 82, no. 7, pp. 1177–1193, 2009.

[55] Luo L., Chakraborty N., Sycara K., "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks", Robotics, IEEE Transactions on 31.1, pp. 19–30, 2015.

[56] LIU X., "Introduction to longitudinal data analysis in psychiatric research". Shanghai Arch Psychiatry, 27(4): 256–259, agosto 2015.

[57] LIU X., "Methods and Applications of Longitudinal Data Analysis", Elsevier, ISBN: 978-0-12-801342-7, 2016.

[58] Lukic M., Barnawi A., Stojmenovic I.. "Robot Coordination for Energy-balanced Matching and Sequence Dispatch of Robots to Events", IEEE Transactions on Computers 64.5, pp. 1416–1428, 2015.

[59] Maus H., van der Aalst M.P., Rickayzen A., Riss U. V. "Challenges for Business Processes and Task Management," Journal of Universal Knowledge Management. Volume 0, Issue 2, 2005.

[60] MDX Language reference, URL: http://msdn.microsoft.com/es-es/library/ms145595.aspx. Último acceso: marzo 2017.

[61] Mens T., Gorp P. V., "A taxonomy of model transformation", ENTCS, vol. 152, pp. 125–142, marzo 2006.

[62] Miorandi D., Sicari S., Pellegrini F. D., Chlamtac I., "Internet of things: Vision, applications and research challenges", Ad Hoc Netw., vol. 10, no. 7, pp. 1497–1516, 2012.

[63] Mohagheghi P., Dehlen V., Neple T., "Definitions and approaches to model quality in model–based software development – A review of literature," Inform. Softw. Technol., vol. 51, no. 12, pp. 1646–1669, 2009.

[64] Moody D. L., "The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering," IEEE Trans. Softw. Eng., vol. 35, no. 6, pp.

756–779, diciembre 2009.

[65] Munkres J., "Algorithms for the Assignment and Transportation Problems", Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38, marzo 1957.

[66] National Center for Analysis of Longitudinal Data in Education Research (CALDER), "What are Longitudinal Data?", URL: http://www.caldercenter.org/what-are-longitudinal-data. Último acceso: marzo 2017.

[67] OMG, Business Process Model and Notation, BPMN specification. URL: http://www.bpmn.org/. Último acceso: marzo 2017.

[68] OMG, Model Driven Architecture –MDA-. URL: http://www.omg.org/mda/. Último acceso: marzo 2017.

[69] OMG, Unified Modeling Language™ (UML®). URL: http://www.omg.org/spec/UML/. Último acceso: marzo 2017.

[70] Pentaho, Mondrian Project, Mondrian Documentation, URL: http://mondrian.pentaho.org/documentation/schema.php. Último acceso: marzo 2017.

[71] Pérez B., Porres I., "Authoring and verification of clinical guidelines: A model driven approach", J. Biomed. Inform., vol. 43, no. 4, pp. 520–536, 2010.

[72] PostgreSQL Database Management System, URL: https://www.postgresql.org/. Último acceso: marzo 2017.

[73] Raidl G. R., Puchinger J., "Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization", Hybrid Metaheuristics: An Emerging Approach to Optimization, Springer Berlin Heidelberg, pp. 31-62, ISBN: 978-3-540-78295-7, 2008.

[74] Rebuge A., Ferreira D. R., "Business process analysis in healthcare environments: A methodology based on process mining," Inform. Syst., vol. 37, no. 2, pp. 99–116, abril 2012.

[75] Roddick J. F., "A survey of schema versioning issues for database systems", Inform. Softw. Technol., vol. 37, no. 7, pp. 383–393, 1995.

[76] Rozinat A., van der Aalst W. M. P., "Conformance checking of processes based on monitoring real behavior," Inform. Syst., vol. 33, no. 1, pp. 64–95, 2008.

[77] Said I. B., Chaâbane M. A., Andonoff E., "A model driven engineering approach for modelling versions of business processes using BPMN", in Proc. BIS, Berlin, Germany, pp. 254–267, 2010.

[78] Scheer A.-W., von Scheel H., von Rosing M., "The Complete Business Process Handbook Volume 1: Body of Knowledge from Process Modeling to BPM", Elsevier, ISBN 0128028602, diciembre 2014.

[79] Simile Projects, Timeline. Web Widget for Visualizing Temporal Data, URL: http://www.simile-widgets.org/timeline/. Último acceso: marzo 2017.

[80] Simmhan Y., Plale B., Gannon D., "A survey of data provenance in e-science", SIGMOD Rec., vol. 34, no. 3, pp. 31–36, 2005.

[81] Sindhgatta R., Ghose A., Dam H.K., "Context-Aware Analysis of Past Process Executions to Aid Resource Allocation Decisions", Advanced Information Systems Engineering, CAiSE 2016, Lecture Notes in Computer Science, vol 9694, Springer, 2016.

[82] SMOTY: Un Sistema de Seguridad Basado en Inteligencia Emergente en el Internet de las Cosas. Ministerio de Ciencia e Innovación, IPT-2011-1328-390000, Logica S. A., Infozara S. L., Complusoft S. L. Universidad de Zaragoza, Universidad de Alcalá, Universidad Carlos III de Madrid. 2011-2014.

[83] The Apache™ FOP Project, URL: https://xmlgraphics.apache.org/fop/. Último acceso: marzo 2017.

[84] THOFU: Tecnologías del Hotel del Futuro, CEN-20101019, Subprograma de Investigación de apoyo a Consorcios Estratégicos Nacionales de Investigación Técnica (CENIT); CDTI: Centro para el Desarrollo Tecnológico Industrial. 2010-2013.

[85] Tsalatsanis A., Yalcin A., Valavanis K. P., "Dynamic task allocation in cooperative robot teams", Robotica 30.05, pp. 721–730, 2012.

[86] Turban E., Aronson J. E., Liang T. P., Sharda R., "Decision Support and Business Intelligence Systems", 8th ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[87] van der Aalst W. M. P., "Exploring the CSCW spectrum using process mining," Adv. Eng. Inform., vol. 21, no. 2, pp. 191–199, 2007.

[88] van der Aalst W., Weijter A., Maruster L., "Workflow mining: Discovering process models from event logs," IEEE Trans. Knowl. Data Eng., vol. 16, no. 9, pp. 1128–1142, septiembre 2004.

[89] Von Neuman J., "A certain zero-sum two-person game equivalent to the optimal assignment problem", Contributions to the Theory of Games 2, pp. 5–12, 1953.

[90] W3C, PROV Model Primer, abril 2013. URL: https://www.w3.org/TR/prov-primer/. Último acceso: marzo 2017.

[91] W3C standards, World Wide Web Consortium (W3C), URL: https://www.w3.org/standards/. Último acceso: marzo 2017.

[92] W3Schools, URL: https://www.w3schools.com/. Último acceso: marzo 2017.

[93] WebSphere Commerce foundation tag library –WCF-, IBM, URL: https://www.ibm.com/support/knowledgecenter/SSZLC2_6.0.0/com.ibm.commerce.component-services.doc/refs/rwvtaglib.htm. Último acceso: marzo 2017.

[94] Weske M., "Business Process Management: Concepts, Languages, Architectures". Second Edition. Springer. 2012

[95] Wright D. R., "Finite State Machines", CSC216, NC State University, 2005. URL: http://www4.ncsu.edu/~drwrigh3/docs/courses/csc216/fsm-notes.pdf. Último acceso: marzo 2017.