UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de fin de carrera de Ingeniero Informático

Implementación, despliegue y estudio de la eficiencia de una librería de generación automática de preguntas

Alberto Herreros García

Directores: Manuel Luque Gallego
José Manuel Cuadra Troncoso
Codirector: Ricardo Conejo Muñoz

Curso: 2013/2014 (3ª convocatoria, marzo 2014)

IMPLEMENTACIÓN, DESPLIEGUE Y ESTUDIO DE LA EFICIENCIA DE UNA LIBRERÍA DE GENERACIÓN AUTOMÁTICA DE PREGUNTAS

Proyecto de fin de carrera de modalidad oferta específica

Realizado por: ALBERTO HERREROS GARCÍA (firma)

Dirigido por: MANUEL LUQUE GALLEGO (firma)

Supervisado por: JOSÉ MANUEL CUADRA TRONCOSO (firma)

Co-Dirigido por: RICARDO CONEJO MUÑOZ (firma)

Tribunal calificador:

Presidente: D./Da. ..... (firma)

Secretario: D./Da. ..... (firma)

Vocal: D./Da. ..... (firma)

Fecha de lectura y defensa: .....

Calificación: .....

Resumen

El presente proyecto de fin de carrera consiste en la implementación, despliegue y estudio de la eficiencia de una librería de generación automática de preguntas en el lenguaje Java. El tema o dominio para la generación de dichas preguntas serán los fundamentos de las redes bayesianas, concretamente los Modelos Gráficos Probabilistas.

La funcionalidad proporcionada por la librería desarrollada nos permitirá confeccionar exámenes o ejercicios de autoevaluación tipo test en formato PDF (Portable Document Format). Por otro lado, la librería se desplegará y formará parte del sistema *Siette* (Sistema de Evaluación inteligente mediante test). En dicho sistema, dispondremos de una asignatura en la que estarán definidas una serie de preguntas que nos servirán para conformar los test de autoevaluación que serán realizados por los alumnos y usuarios del sistema.

Se realizará un estudio sobre la eficiencia de la librería tomando los tiempos que tarda la generación de cada tipo de pregunta. Dicho estudio nos permitirá concluir que la librería implementada es apta para su uso dentro del sistema *Siette,* como herramienta útil para la autoformación de los alumnos en el dominio del tema Modelos Gráficos Probabilistas.

Lista de palabras clave

- Test
- Librería
- Probabilidad
- Variable aleatoria
- Redes bayesianas
- Heurística
- Pregunta
- Sistema
- Distribución de probabilidad
- Grafo
- Examen
- Asignatura
- Inferencia
- Hallazgo
- Evidencia
- SIETTE

Abstract

This project consists in developing an automatic questions generation library in Java language. The project consists of implementation, deployment and efficiency study of the library. The subject or domain for question generations are the basics of Bayesian networks and have to do with Probabilistic Graphical Models.

The functionality of the library can be using in order to make exams or evaluations tests in PDF (Portable Document Format). On the other hand, the library can be deployment as part of *Siette* platform.

A subject about Probabilistic Graphical Models will be defined in *Siette* system as a set of multiple choice questions that will be the basics for test generations to be made for users systems and students.

An efficiency study will be carry out in order to take the spending time in generation for each type of questions in the library. As the result of this study, we can conclude that the deployed library can be used as part of *Siette* system like a powerful training tool for students and autolearning in the subject domain about Probabilistic Graphical Models.

List of keywords

- Test
- Library
- Probability
- Random variable
- Bayesian networks
- Heuristics
- Question
- System
- Probability distribution
- Graph
- Exam
- Subject
- Inference
- Finding
- Evidence
- SIETTE

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La Universidad Nacional de Educación a Distancia (UNED) es la universidad con más alumnos de España y su número va en aumento cada curso. Las nuevas tecnologías están ayudando a una rápida extensión de la docencia a distancia a través de Internet, que incluso está siendo adoptada parcialmente por universidades de tipo presencial. Por otro lado, el Espacio Europeo de Educación Superior (EEES) está haciendo que se haga un especial énfasis en el aprendizaje activo en la enseñanza, donde el alumno realice gran cantidad de actividades y trabajos de manera autónoma, ayudado por la flexibilidad que aporta el estudiar a distancia desde cualquier lugar en cualquier momento del día, promviendo una teoría basada en el supuesto de que sólo se logra un aprendizaje eficaz cuando es el propio alumno el que asume la responsabilidad en la organización y desarrollo de su trabajo académico (de Miguel Díaz, 2007). Estos hechos (el número creciente de alumnos de la UNED junto con un mayor número de actividades a realizar por cada uno de ellos) están haciendo que a los equipos docentes de las asignaturas les resulte cada vez más difícil y les lleve un mayor tiempo crear actividades para los alumnos que permitan que éstos consigan los objetivos del aprendizaje de una manera más eficaz. Es por ello que hemos visto como necesaria la existencia de herramientas software que permitan, al menos parcialmente, la generación automática de actividades para los alumnos. En este contexto, el presente proyecto de fin de carrera pretende realizar una aportación más para conseguir el objetivo de disponer de una serie de recursos didácticos, en forma de herramientas software, que puedan servir al profesor para facilitar la tarea de confeccionar ejercicios o test de autoevaluación para los alumnos en la docencia de una asignatura y que los alumnos sean capaces, con ayuda de estas herramientas, de enriquecer y acelerar

su proceso de aprendizaje de una manera autónoma.

1.2. Objetivos

El objetivo del presente proyecto es construir una librería de generación automática de preguntas de test que sirvan para la autoevaluación y como herramienta de apoyo en la docencia de asignaturas que traten o manejen conceptos básicos sobre el tema Modelos Gráficos Probabilistas (F.J.Díez, 2010). El presente proyecto de fin de carrera se enmarca en el cumplimiento de las directrices de la declaración de Bolonia para un Espacio Europeo de Educación Superior, en el que están involucradas actualmente las universidades españolas. Igualmente se enmarca en el esfuerzo por poner en marcha distintos tipos de herramientas de formación práctica y a distancia basados en aprendizaje colaborativo o en aprendizaje de alto rendimiento basado en tareas reales. El objetivo principal que se pretende alcanzar con la realización de este proyecto es que la librería desarrollada llegue a estar integrada en el sistema ***Siette*** [1] y sirva a dicho sistema como abastecimiento de preguntas de test para la asignatura de Modelos Gráficos Probabilistas. *Siette* es un sistema web que permite la creación y mantenimiento de bancos de preguntas, y realización de tests, y que implementa la Teoría Clásica de Test (CTT), Teoría de Respuesta al Ítem (TRI), y permite realizar Tests Adaptativos Informatizados (TAI), y puede usarse como herramienta para el aprendizaje colaborativo. Además puede usarse como módulo de evaluación de un Sistema Tutor Inteligente (STI) o conectado a un Sistema Gestor de Contenidos Educativos (LMS) como *Moodle*. La librería desarrollada en el presente proyecto, abastecerá al sistema *Siette* de una batería de preguntas de test en el dominio de una asignatura más, a partir de la cuál, se podrán plantear distintos tipos de preguntas y confeccionar diversos tipos de test por parte del equipo docente para poder ser realizados por los alumnos y usuarios del sistema. El presente proyecto consiste en la implementación, despliegue y estudio de la eficiencia de una librería de generación automática de preguntas, en el contexto del tema **Modelos Gráficos Probabilistas** y dentro de la asignatura **Métodos Probabilistas en Inteligencia Artificial**.

La integración con el sistema *Siette* nos permitirá probar la librería implementada y realizar un estudio sobre la eficiencia de la misma tomando los tiempos que tarda en la generación de cada pregunta, así como evaluar la calidad en la generación automática y aleatoria de la parte cambiante de la pregunta. Se valorará también la información de refuerzo que aporta la librería cuando no se de una respuesta válida a la pregunta

[1] 1 Consúltese información sobre Siette en `http://www.siette.org/siette/`

planteada o se conteste erróneamente o para el caso que no se de ninguna respuesta.

A lo largo de la presente memoria, se presentaran los distintos tipos de preguntas que se pueden plantear en este proyecto, introduciendo brevemente la base teórica o fundamentos sobre la que se sustenta cada una de ellas y explicando cómo se ha realizado la implementación del programa de generación automática de cada tipo de pregunta, identificando y distinguiendo su parte fija de la parte variable que se genera de manera automática haciendo uso de métodos heurísticos combinados con generación aleatoria. Se seguirán las etapas clásicas de la ingeniería del software propias del proceso de desarrollo de software.

Además de su integración con el sistema *Siette*, la librería implementada permitirá generar preguntas de tipo test para ejercicios de autoevaluación o bien para la generación de exámenes de tipo test en formato PDF (*Portable Document Format*) y podrá ser utilizada desde un programa cliente en Java que haciendo uso de la librería desarrollada, permita generar exámenes de test en formato PDF combinando varias preguntas de las que se ofertan en la librería.

1.3. Estructura de la memoria

En este apartado vamos a explicar la estructura que se ha seguido para la elaboración de esta memoria del proyecto.

En este primer capítulo se realiza una breve introducción al proyecto, explicando la motivación del mismo y especificando los objetivos que se pretenden alcanzar tras su realización.

Tras el capítulo introductorio, en el capítulo 2, se discute la metodología más adecuada a emplear para el desarrollo de proyectos de este tipo como es un proyecto de fin de carrera de ingeniería informática.

El capítulo 3 presenta la planificación del proyecto sobre un diagrama de *Gantt,* identificando los requisitos de usuario del proyecto tanto los funcionales como los no funcionales. Se identifican las tareas a llevar a cabo en cada una de las etapas del ciclo de vida del proyecto para finalmente realizar un estudio de viabilidad del mismo.

En el cuarto capítulo se abordan los fundamentos teóricos de los bloques temáticos de las preguntas que se generan de manera automática por la librería desarrollada. Estos fundamentos son la materia de estudio de asignaturas que tienen que ver con la teoría sobre los Modelos Gráficos Probabilistas y que constituyen el conocimiento que un alumno debe tener para resolver las preguntas que se plantean en esta librería de generación automática.

El capítulo 5 está dedicado al análisis del problema a resolver, explica como se va a afrontar la implementación de la librería, dividiendo el desarrollo de la misma en tres partes que se corresponden con los tres bloques temáticos que se van a tratar en la generación automática de las preguntas y presentando los distintos actores y casos de uso del sistema, componentes y preguntas tipo.

Una vez superada la fase de análisis del proyecto, en el capítulo 6 se explica el diseño de la solución adoptada para la implementación de la librería de generación automática de preguntas.

El capítulo 7 realiza una exposición de los distintos componentes software desarrollados en forma de clases y paquetes Java que forman parte de la librería implementada. Se comentarán las clases principales con sus métodos más representativos, así como las posibilidades de ampliación de la librería con nuevos componentes que nos permitan generar nuevos tipos de preguntas para la generación automática.

El capítulo octavo presenta los distintos tipos de test de unidad que se han definido para la fase de pruebas del proyecto, cada tipo de pregunta tendrá asociado su correspondiente test en *JUnit* para validar la generación de cada tipo de pregunta. Por otro lado se realizarán test de eficiencia de la generación de cada pregunta, para ello se presentan en este capítulo los distintos test que se han definido para evaluar la eficiencia en la generación de cada tipo de pregunta.

Por último en el capítulo 9 se comentan las conclusiones alcanzadas en el presente proyecto y las líneas de trabajo futuro a seguir así como las posibilidades de ampliación del mismo.

En la última parte de la memoria se presentan una serie de apéndices y que son los siguientes:

- En el apéndice A se identifican y enumeran los distintos tipos de preguntas del repertorio ofrecido por la librería implementada.
- El apéndice B presenta el manual de usuario de la librería desarrollada.
- El apéndice C contiene el manual de instalación.
- En el apéndice D se presenta un ejemplo de test generado con la librería desarrollada en este proyecto para cada uno de los bloques temáticos considerados.
- Por último, en el apéndice E, se presenta un listado con las distintas siglas, abreviaturas y acrónimos que se han utilizado en el presente proyecto.

Capítulo 2

Metodología elegida

2.1. Metodología elegida. Justificación

Entre los distintos modelos de ciclo de vida que se usan tradicionalmente en la ingeniería del software (modelo en cascada, modelo en V, uso de prototipos, modelo en espiral, incremental, evolutivo, etc...) (S.Pressman, 2003) se ha elegido el Ciclo de Vida en Cascada (ver figura 2.1) para la realización del presente proyecto de fin de carrera. Las razones de esta elección se justifican por ser el modelo que se cree mejor se adapta al problema a resolver, debido a que se observan las siguientes características que hacen recomendable dicha elección:

1. Los requisitos son estables, están bien definidos y bien comprendidos.
2. El diseño y la tecnología está probada y madura.
3. La duración del proyecto será relativamente corta en el marco de lo que suele ser un proyecto de ingeniería del software.[1]
4. El usuario no necesita versiones intermedias del producto.[2]

Por todas estas razones expuestas en los puntos anteriores podemos clasificar este proyecto como un proyecto clásico de ingeniería del software que va a dar solución a un problema de la vida real que en este caso se trata de generar una herramienta didáctica para el aprendizaje y evaluación de conocimientos adquiridos acerca de una materia concreta.

[1] La duración del proyecto se alarga en el tiempo debido a que el equipo de trabajo está compuesto única y exclusivamente por una sóla persona que, en este caso, es el alumno que lleva a cabo el proyecto fin de carrera.

[2] La librería se puede dividir, no obstante, en distintos tipos de preguntas sobre varios bloque temáticos lo que podría dar lugar a versiones intermedias del producto y consecuentemente a distintos entregables.

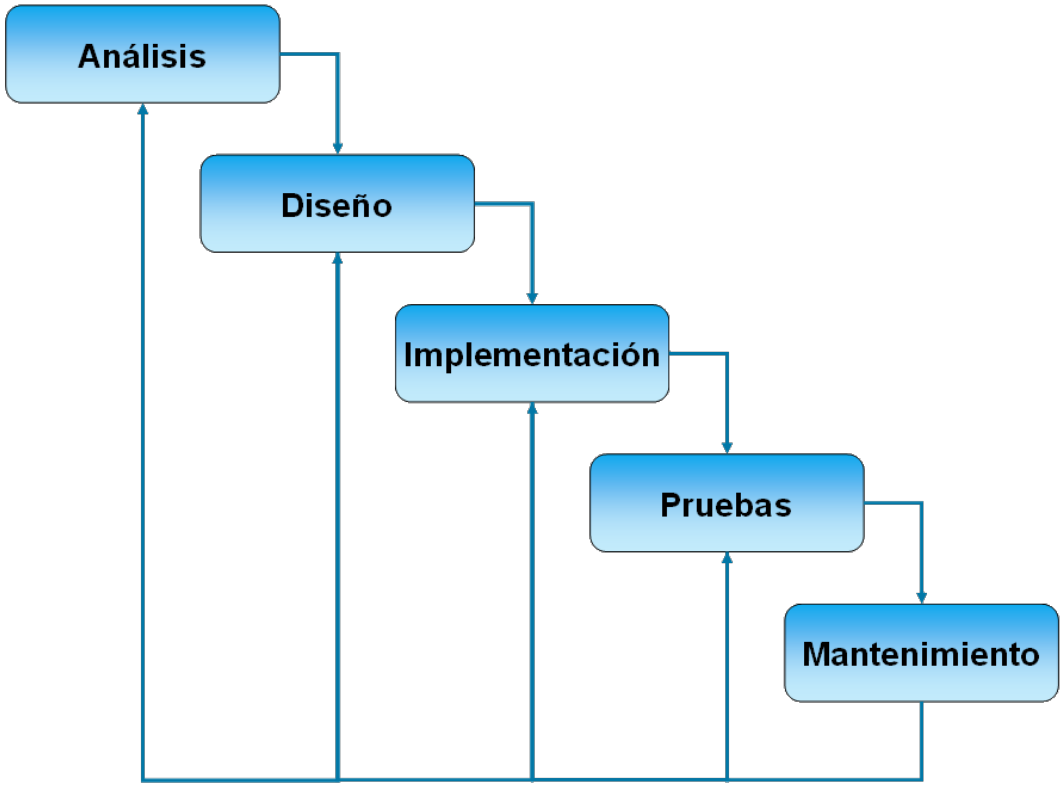


Figura 2.1: Modelo de ciclo de vida en cascada.

2.2. Fases de la ingeniería del software

Como se ha comentado en el apartado anterior, para la realización del presente proyecto se seguirá el método de desarrollo clásico de la ingeniería del software, siguiendo el modelo clásico o en cascada que incluye las siguientes etapas:

- Definición de requisitos del usuario.
- Análisis de requisitos.
- Diseño de la solución.
- Codificación o implementación de la librería.
- Pruebas unitarias y de integración con sistema *Siette*.
- Documentación del software desarrollado y redacción de la memoria del proyecto.
- Implantación y despliegue en sistema *Siette*.

Para el desarrollo de la librería se utilizarán técnicas propias de la Programación Orientada a Objetos (*POO*), proporcionada por las propias características del lenguaje de programación Java. El paradigma de programación orientada a objetos facilita importantes características que serán indispensables en el desarrollo de la librería: abstracción, encapsulamiento, principio de ocultación, herencia y polimorfismo. La orientación a objetos es el paradigma de programación más utilizado actualmente en el desarrollo de aplicaciones. Esto se debe a la existencia de lenguajes como Java, C# y otros que aportan otras muchas características que los hacen fáciles de usar, potentes, compatibles con muchas plataformas y sobre todo seguros y fáciles de depurar.

El paradigma de programación orientada a objetos establece una asociación entre la información y las sentencias encargadas de gestionar dicha información. Desde un punto de vista de la programación procedimental, la orientación a objetos se puede ver como la asociación entre funciones y procedimientos a los tipos de datos que manejan(S.Pressman, 2003). Las principales aportaciones de este paradigma al desarrollo de sistemas software son la herencia y el polimorfismo. La herencia es una característica que permite construir una clase (conocida como clase hija) basándose o teniendo como plantilla otra clase construida previamente (conocida como clase padre). Al utilizar una clase padre como base para definir una clase hija los atributos y métodos definidos en la clase padre están disponibles en la clase hija. Es decir, los objetos de la clase hija tendrán valores para los atributos definidos en la clase hija y también tendrán valores para los atributos definidos en la clase padre. Esta característica permite una mayor reutilización del código, ya que si dos clases de un programa tienen ciertas características en común, se puede construir una clase que sea clase padre de las dos clases que alberga los atributos y métodos que sean comunes a ambas. El polimorfismo es una característica de la programación orientada a objetos, muy relacionada con la herencia, que solventa un conjunto de problemas típicos que aparecían en la programación procedimental. En multitud de aplicaciones es necesario gestionar información con distinta estructura de una forma homogénea. Veremos estas dos características aplicadas en este proyecto al definir, por ejemplo, las clases que modelan las relaciones de independencia probabilística entre variables aleatorias de un sistema.

En la parte de diseño se utilizará el lenguaje de modelado UML (*Unified Modeling Language*) (C.Martin, 2004), concretamente los diagramas de clases (estructura) y los diagramas de casos de uso (comportamiento), así como los diagramas de componentes que nos proporcionarán una visión global del sistema implementado.

Capítulo 3

Gestión del Proyecto

3.1. Especificación de Requisitos

La captura o especificación de requisitos es una parte fundamental en el proceso de desarrollo de software y la parte más importante en la etapa de definición de un proyecto de ingeniería de software (Agustín, 2003). Un proyecto de ingeniería de software se plantea por el hecho de que una organización o una persona tiene una necesidad o problema que se soluciona con el desarrollo de un sistema informático.

En lo que respecta a este proyecto, vamos a considerar que el usuario que presenta la necesidad es el departamento de Inteligencia Artificial de la UNED que oferta o propone la realización de un Proyecto de Fin de Carrera a través de un director asignado al mismo. Para la captura de requisitos del presente proyecto, se parte de un documento inicial donde se explica la motivación del proyecto y los objetivos que han de alcanzarse tras la realización del mismo. De la lectura detenida de dicho documento, se pueden identificar ya una serie de requisitos que se enumeran en la tabla 3.1. Posteriormente se realiza una web conferencia entre el equipo docente y los alumnos interesados, donde se comentan los objetivos del proyecto y las herramientas que deben utilizarse para la realización del mismo. A partir de este momento comienzan una serie de entrevistas o contactos con el director del proyecto para intentar recavar toda la información posible que nos permita abordar con éxito esta primera etapa de toma de requisitos.

Como primera toma de contacto y para familiarizarse con las herramientas que se van a utilizar en el desarrollo del proyecto, el equipo docente propone la realización de una práctica, denominada práctica cero, que permite tener, tras su realización, una idea clara de los objetivos que se pretenden conseguir con el presente proyecto y de las necesidades que se quieren satisfacer con la creación de las herramientas software que permitan realizar

actividades de autoformación para los alumnos.

Con toda la información disponible podemos hacer una primera división de los requisitos entre funcionales y no funcionales. En los siguientes apartados se pasa a catalogar y detallar cada uno de los requisitos del presente proyecto. Muchos de ellos son obtenidos directamente de las indicaciones dadas por el equipo docente y el director del proyecto en el documento inicial y la práctica cero, mientras que otros se han obtenido de las distintas tomas de contacto con el director del proyecto a lo largo de todo el desarrollo.

3.1.1. Requisitos funcionales

En los siguientes apartados se van a identificar y especificar cada uno de los requisitos funcionales que ha de satisfacer la librería implementada. Es decir, se van a definir las funciones que el sistema debe realizar. [1]. La nomenclatura a utilizar se indica a continuación:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.
- palabra ***REQ*** que es la abreviatura de requisito.
- palabra ***FUN*** seguido de un número de orden, que indica que se trata de un requisito funcional y el número que completa su identificación junto con el resto de palabras.

No se va a tener en cuenta la prioridad ni la importancia de los requisitos a la hora de catalogarlos, de manera que todos los requisitos especificados serán de importancia alta y de máxima prioridad. Todos ellos serán implementados en el proyecto sin descartar ninguno por razones de coste o por tiempo de realización. Pasamos a definir a continuación, cada uno de los requisitos funcionales:

LGAP_REQ_FUN1.- **Componentes** El software que desarrolle el alumno tendrá que componerse de dos elementos o componentes principales:

- Una librería (o conjunto de librerías) que permitan a un programa cliente obtener preguntas generadas para la asignatura concreta.
- Una asignatura de prueba en el sistema *Siette* que demuestre que la librería implementada funciona adecuadamente.

[1] La nomenclatura a utilizar consistirá en una serie de palabras separadas por el carácter '_' (guión bajo)

***LGAP_REQ_FUN2*.- Generación automática de preguntas** El principal objetivo del proyecto es la generación automática de preguntas para los alumnos. El sistema deberá generar preguntas cortas de test de manera automática destinadas a que sean respondidas por los alumnos. El sistema desarrollado, más concretamente la librería generada, deberá proporcionar el enunciado de una pregunta obtenido de manera automática. El contenido variable del enunciado se generará aleatoriamente.

***LGAP_REQ_FUN3*.- Respuestas propuestas a cada pregunta planteada** Para cada pregunta generada, se presentarán un número determinado de posibles respuestas a la pregunta planteada llamadas alternativas. Al menos una de las respuestas propuestas ha de ser correcta. Este es un requisito impuesto también en el Sistema *Siette* para todo tipo de preguntas con alternativas ya sea de respuesta simple (sólo una de las alternativas propuestas es la respuesta correcta) o de respuesta múltiple (una o más de las alternativas propuestas puede ser una respuesta correcta).

***LGAP_REQ_FUN4*.- Parametrización en las alternativas planteadas** Las alternativas o respuestas ofrecidas a la pregunta planteada han de poder ser parametrizadas en la librería. El programa que haga uso de la librería y en su caso el Sistema *Siette*, podrá elegir el número de alternativas que se presentan a una pregunta planteada. De cada una de estas alternativas se podrá elegir cuantas de dichas alternativas han de ser respuestas correctas (en los casos en los que pueda existir más de una respuesta correcta) y cuantas serán respuestas incorrectas.

***LGAP_REQ_FUN5*.- Dominio de las preguntas generadas** Las preguntas generadas se ceñirán a un determinado dominio. La asignatura o el dominio elegido para este proyecto de fin de carrera son los **Modelos Gráficos Probabilísticos** (F.J.Díez, 2010). Dentro de este tema, vamos a generar preguntas de manera automática para tres tipos de subdominios que en adelante llamaremos bloques temáticos y que conforman la base teórica o de conocimiento que un alumno debe tener para poder responder a las preguntas que se plantean en esta librería de generación automática de preguntas. A continuación definimos cada uno de los tres bloques temáticos que se van a considerar en este proyecto:

- bloque temático 1.- relaciones de independencia probabilística y de independencia probabilística condicional que se plantean sobre una distribución de probabilidad en un sistema compuesto por un conjunto de variables aleatorias discretas.

- bloque temático 2.- relaciones de independencia probabilística y de independencia

probabilística condicional que se plantean sobre un grafo (dirigido o no dirigido) acíclico.

- bloque temático 3.- Inferencia en Redes Bayesianas. Dada la estructura de una red bayesiana, así como su parte cuantitativa, se introduce uno o varios hallazgos y se pregunta por la probabilidad a posteriori tras la evidencia introducida en alguna variable o nodo de la red.

***LGAP_REQ_FUN6.*- Refuerzo para las respuestas correctas** Deberá proporcionarse algún método en la librería que nos permita obtener el refuerzo positivo para una respuesta dada. Es decir, el sistema tendrá que ser capaz de proporcionar una breve explicación de cómo se ha llegado hasta la respuesta correcta a la pregunta planteada de manera que el alumno pueda ver claramente como debe resolver cada pregunta y como ha razonado el sistema para llegar a esa conclusión.

***LGAP_REQ_FUN7.*- Refuerzo para las respuestas incorrectas** De la misma forma, cuando el alumno marque como respuesta correcta una que no lo es, el sistema ha de proporcionar algún mecanismo que permita explicar porqué la respuesta marcada no es correcta e indique los pasos a seguir para llegar a la solución correcta. De esta manera el alumno podrá identificar dónde ha cometido el error y podrá avanzar en su propio proceso de aprendizaje.

***LGAP_REQ_FUN8.*- Examen en formato PDF** La librería desarrollada deberá poder ser utilizada por un programa cliente en Java y ofrecerá la funcionalidad de poder generar exámenes de tipo test en formato PDF, confeccionado como una batería de preguntas obtenidas del repertorio ofrecido por la librería eligiendo de manera aleatoria cada una de las preguntas de entre los tres bloques temáticos.

***LGAP_REQ_FUN9.*- Examen en formato PDF con solución** La librería desarrollada deberá poder ser utilizada por un programa cliente en Java y ofrecerá la funcionalidad de poder generar exámenes de tipo test en formato PDF, confeccionado como una batería de preguntas obtenidas de la librería de manera aleatoria. Además se indicará cuál de las alternativas o respuestas a la pregunta planteada es correcta por ejemplo poniendo entre paréntesis la palabra “Correcta” tras la opción correcta.

Tabla resumen requisitos funcionales En la figura 3.1 se resumen los requisitos que se han identificado en los apartados anteriores a modo de catálogo y como primera versión

de la matriz de trazabilidad de los requisitos que nos permitirá hacer un seguimiento de los mismos y que se completará en las fases siguientes con los requisitos de diseño, código y caso de prueba correspondientes.

Tabla 3.1: Especificación de requisitos funcionales.

REQUISITOS FUNCIONALES	DESCRIPCIÓN
LGAP_REQ_FUN1	Componentes
LGAP_REQ_FUN2	Generación automática de preguntas
LGAP_REQ_FUN3	Respuestas propuestas
LGAP_REQ_FUN4	Parametrización en alternativas
LGAP_REQ_FUN5	Dominio de las preguntas generadas
LGAP_REQ_FUN6	Refuerzo respuestas correctas
LGAP_REQ_FUN7	Refuerzo respuestas incorrectas
LGAP_REQ_FUN8	Examen formato PDF
LGAP_REQ_FUN9	Examen formato PDF con solución

3.1.2. Requisitos no funcionales

Una vez especificados los requisitos funcionales del sistema, se especifican los no funcionales:

LGAP_REQ_NOFUN1.- **Escalabilidad** Para darle al sistema características de herramienta de colaboración se establece que el sistema debe ser fácilmente escalable. Un estudiante de ingeniería en informática sin demasiada especialización debe ser capaz de colaborar en la ampliación y mejora del sistema desarrollado.

LGAP_REQ_NOFUN2.- **Fiabilidad** Las preguntas generadas han de ser fiables, se decir han de tener lógica y ser planteadas sin ambigüedades; han de ser entendibles por profesores, alumnos y el resto de usuarios de la librería.

LGAP_REQ_NOFUN3.- **Disponibilidad** Se ha de garantizar su disponibilidad en el marco de su integración con el sistema *Siette*.

LGAP_REQ_NOFUN4.- **Mantenibilidad** Para garantizar la mantenibilidad de la librería desarrollada, ésta debe estar implementada de la forma más clara y comprensible posible, utilizando para ello las normas de estilo propias de la comunidad de programadores Java y siguiendo el paradigma de la *POO*, patrones de diseño, etc.(Shalloway and

Trott, 2004) Además la librería debe estar lo suficientemente bien documentada para que un programador pueda comprender el código y modificarlo lo más rápidamente posible y definir nuevos tipos de preguntas en el dominio propuesto o en otros dominios.

***LGAP_REQ_NOFUN5.*- Portabilidad** La librería será implementa en el lenguaje de programación Java para aprovechar las cualidades respecto a la portabilidad de esta plataforma. Se establece la necesidad del diseño de las herramientas didácticas para su utilización en los entornos mayoritarios de ejecución (al menos Linux y Windows).

***LGAP_REQ_NOFUN6.*- Rendimiento. Tiempo de respuesta** Deberá garantizarse que los algoritmos y procesos implementados en la librería tengan buenos tiempos de respuesta. Según se indica en el título del proyecto, uno de los principales objetivos del presente proyecto es estudiar la eficiencia de la librería. Un estudio de la eficiencia de la librería que muestre estadísticas de tiempo (tanto el caso medio como el peor caso), que garantice un rendimiento aceptable cuando sea invocada en el entorno del sistema *Siette*.

***LGAP_REQ_NOFUN7.*- Software Libre** Un requisito de este proyecto fin de carrera es una clara apuesta por el software libre. A pesar de la recomendación realizada por el equipo docente de utilizar el sistema operativo *Linux*, se ha optado por utilizar el sistema *Windows* en su lugar. El resto de las herramientas necesarias para el desarrollo del presente proyecto se han instalado en una máquina virtual definida sobre dicho sistema [2] y son todas ellas de software libre.

***LGAP_REQ_NOFUN8.*- Pruebas y entorno recomendado** Se utiliza como entorno integrado de desarrollo *Eclipse for Java Developers (Kepler)* con el *jdk 1.7*, *Javadoc* para la documentación de la librería y *JUnit 4.8.2* para las pruebas unitarias.

El programa de generación de preguntas tiene que generar las preguntas en el formato *Latex*, ya que sirve para generar salida a PDF, y, además, la misma salida en *Latex* junto con *Mathjax* serán utilizadas al hacer el despliegue en el sistema *Siette*.

La presente memoria del proyecto se redacta también en *Latex*, a través del editor de alto nivel *Lyx* versión 2.0.3.

Como sistema de control de versiones en la nube se utiliza *Bitbucket*. Todo el software del proyecto está publicado en el repositorio de proyecto de *Bitbucket* siguiente: https://bitbucket.org/aherrerosg/es.uned.dia.pfc.lgap.mgp.

[2] Microsoft Windows XP Professional versión 2002 Service Pack 3

***LGAP_REQ_NOFUN9.-* Herramientas y Software a utilizar** Todas las herramientas software que se utilizan para la realización del presente proyecto son libres o de código abierto. A continuación se enumera cada una de ellas:

- ***ArgoUML***: herramienta para la generación de diagramas UML.
  (ArgoUML,)
- ***BitBucket***: servicio de alojamiento web para almacenar y realizar la gestión de configuración de todos los recursos software utilizados en el presente proyecto: fuentes, documentación, memoria, etc. (bitbucket,)
- ***Eclipse***: entorno integrado de desarrollo para la librería Java a desarrollar en el proyecto. (Foundation,)
- ***MikTex***: distribución Latex para Microsoft Windows. (Schenk, 2013)
- ***InkScape***: editor de gráficos vectoriales de código abierto para edición y manipulación de gráficos en formato SVG.(Source,)
- ***Lyx***: procesador de documentos de código abierto para la redacción de la memoria del proyecto y de toda la documentación asociada al mismo. (Lyx,)
- ***MathJax***: motor Javascript para el uso de expresiones matemáticas en páginas web. (MathJax, 2011)
- ***OpenMarkov***: herramienta para la generación y evaluación de modelos gráficos probabilistas que en este proyecto se usará como API para la inferencia en redes bayesianas [3].
  (UNED, 2013)
- ***OpenProj***: herramienta para la planificación de proyectos. (ope,)
- ***Siette***: sistema web que permite la creación y mantenimiento de bancos de preguntas, y realización de tests, y que implementa la Teoría Clásica de Test (CTT), Teoría de Respuesta al Ítem (TRI), y permite realizar Tests Adaptativos Informatizados (TAI), y puede usarse como herramienta para el aprendizaje colaborativo. Además puede usarse como módulo de evaluación de un Sistema Tutor Inteligente (STI) o conectado a un Sistema Gestor de Contenidos Educativos (LMS) como *Moodle*.
  (Siette, 2013)

[3] Centro de Investigación sobre Sistemas Inteligentes de Ayuda a la Decisión Universidad Nacional de Educación a Distancia (UNED)

- ***SVG***: especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en formato XML. (SVG,)

- ***Tomcat***: contenedor de *Servlets* y *JSP* para las pruebas de despliegue de la librería desarrollada antes de su integración en el sistema Siette. (Apache, 2011)

Tabla resumen requisitos no funcionales En la figura 3.2 se catalogan los requisitos no funcionales identificados y descritos en los apartados anteriores.

Tabla 3.2: Especificación de requisitos no funcionales.

REQUISITOS NO FUNCIONALES	DESCRIPCIÓN
LGAP_REQ_NOFUN1	Escalabilidad
LGAP_REQ_NOFUN2	Fiabilidad
LGAP_REQ_NOFUN3	Disponibilidad
LGAP_REQ_NOFUN4	Mantenibilidad
LGAP_REQ_NOFUN5	Portabilidad
LGAP_REQ_NOFUN6	Rendimiento
LGAP_REQ_NOFUN7	Software Libre
LGAP_REQ_NOFUN8	Pruebas
LGAP_REQ_NOFUN9	SW a utilizar

Para realizar la gestión del presente proyecto se seguirán las directrices marcadas en la literatura de gestión de proyectos de desarrollo de software y de la propia ingeniería del software (Sommerville, 2005). Una vez superada la primera fase de este proyecto, explicada en el apartado anterior, en la que se han identificado una lista de requisitos tanto funcionales como no funcionales que tiene que satisfacer el sistema para dar solución a una necesidad concreta del usuario, en esta fase del proyecto vamos a identificar cada una de las tareas a llevar a cabo a lo largo del proyecto y a partir de estas, establecer una planificación de las mismas sobre un diagrama de Gantt.

En un proyecto de ingeniería de desarrollo de software, una vez identificados los requisitos, se construye un equipo de trabajo del proyecto definiendo y asignando una serie de roles: jefe de proyecto, analistas, programadores, etc. Un proyecto de Fin de Carrera como el que nos ocupa no es un proyecto de trabajo en equipo sino una actividad totalmente individual. Por tanto a la hora de establecer la planificación del proyecto sólo se tendrán en cuenta las tareas a realizar, siendo el único recurso disponible el propio alumno que

realiza el proyecto que será el realizador de todas las tareas y asumirá por tanto los distintos roles que se requieren en la realización de cada una de ellas en las distintas etapas del proyecto.

La gestión del proyecto se realiza a lo largo de todo el ciclo de vida del software desde el principio cuando se realiza la asignación del proyecto y su director hasta el día de la defensa del mismo ante el tribunal. Según el modelo elegido tendremos las distintas fases del proyecto (ver apartado2.2). En cada una de estas fases vamos a realizar un desglose de las distintas actividades o tareas que se llevarán a cabo en cada una de las etapas. En el apartado anterior ya se ha realizado un catálogo de requisitos de usuarios, funcionales y no funcionales. En los siguientes apartados, se continuará realizando el desglose de tareas de cada una de las fases hasta tener la planificación completa del proyecto y la duración estimada del mismo. Dejaremos a un lado el coste económico del proyecto considerando únicamente como coste el tiempo invertido en la realización del mismo.

3.2. Identificación de tareas

Teniendo siempre presente los requisitos obtenidos en el apartado anterior, en los apartados siguientes procederemos a la identificación de las tareas a llevar a cabo a lo largo del proyecto en cada una de las etapas del ciclo de vida del mismo.

3.2.1. Tareas de Análisis

La nomenclatura que se va a utilizar a partir de ahora para definir las tareas de análisis consistirá en una serie de palabras separadas por el carácter guión bajo. A continuación se especifica la nomenclatura a utilizar para las tareas de análisis:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.
- palabra ***ANA*** que es la abreviatura de análisis para indicar que se trata de una tarea de análisis.
- la siguiente palabra describe la acción que se lleva acabo con la ejecución de cada tarea.

El primer paso en nuestro análisis será definir una base o componente común sobre la que se va a construir el resto de la librería. Esta tarea la identificamos en el diagrama como ***LGAP_ANA_COMUN***. Aquí analizaremos la parte común que caracteriza a

una pregunta de test y las funcionalidades básicas necesarias para el posterior desarrollo, como pueden ser generaciones aleatorias de números, algoritmos comunes, combinatoria, formateo de la salida a generar, excepciones, logs de la aplicación, etc.

El siguiente paso, una vez analizada la parte común, es empezar a definir cada bloque temático y, dentro de cada bloque temático, cada tipo de pregunta distinta que se va a plantear. Realizaremos una serie de iteraciones en el desarrollo de la librería cada vez que incorporamos a la misma un nuevo bloque temático o una nueva pregunta o tipo de pregunta dentro de un bloque temático. Los pasos que se siguen en este proceso son los siguientes:

1. Elegir el tipo de pregunta a plantear que estará enmarcada en un determinado bloque temático de los definidos en el dominio (ver 3.1.1). A partir del conocimiento obtenido con los apuntes y la documentación proporcionada por el director del proyecto acerca de la asignatura **Métodos Gráficos Probabilistas en Inteligencia Artificial** (F.J.Díez, 2010).

2. Realizar el análisis de cada tipo de pregunta. Este análisis incluye la resolución sobre el papel de la pregunta en cuestión que dará como resultado el establecimiento de un plan que nos permita llegar a la solución a la pregunta y su posterior diseño e implementación en el ordenador.

3. Realizar el diseño de la solución, incorporando los diagramas de clases en UML necesarios: diagrama de clases, casos de uso, componentes, etc.

4. Codificación en lenguaje Java y haciendo uso de las técnicas de *POO* del diseño realizado en el punto anterior. Se incluyen aquí también el uso de patrones de diseño típicos en la comunidad de desarrolladores como el patrón *Singleton* que se utilizar en la clase *ExamenTest* como contenedor de preguntas de test, como se verá más adelante.

5. Pruebas unitarias de cada tipo de pregunta a generar. Para ello se definirá el correspondiente test en *JUnit* y se validará la salida a consola en primera instancia, así como la salida *Latex* de la pregunta generada de forma individual. También se realiza la prueba correspondiente de generación del tipo de pregunta desde la página JSP de prueba, probando cada una de las posibilidades de parametrización que se ofrecen y que darán lugar a la variedad en las preguntas que se generan en el repertorio ofrecido por la librería. Esta página JSP nos servirá para probar la librería

desde un programa cliente y cumplir con uno de los requisitos establecidos en el proyecto.

6. Integración en el sistema *Siette*. Definiremos en *Siette*, a partir de las posibilidades de parametrización que ofrece cada tipo de pregunta, las distintas instancias que darán lugar a las preguntas finales que se utilizarán para definir la asignatura en *Siette* y posteriormente los test de autoevaluación correspondientes (ver apéndice A) que serán realizados por los alumnos y usuarios del sistema.

7. Estudio de la eficiencia en la generación de cada pregunta tomando el tiempo más corto y más largo en la generación de cada pregunta de manera que se garantice un buen tiempo de respuesta dentro de su integración en el sistema *Siette* y constituya por tanto un sistema útil para alumos que cursen o manejen los conceptos básicos sobre la asignatura definida.

En la organización que se ha considerado en nuestra librería, ésta contendrá un total de tres bloques temáticos cuyo desarrollo iremos viendo por fases en los apartados siguientes del ciclo de vida del desarrollo. Cada uno de estos tres bloques temáticos dará lugar a distintos tipos de preguntas. Así tendremos:

- **bloque temático 1:** relaciones de independencia probabilística y de independencia probabilística condicional a partir de una distribución de probabilidad dada sobre un conjunto de variables aleatorias que componen el sistema. Este bloque será una de las bases para la posterior construción de una red bayesiana, constituyendo la parte cuantitativa de la misma.

- **bloque temático 2:** relaciones de independencia probabilística y de independencia probabilística condicional en grafos. Este bloque será la base que complementa la definida en el punto anterior para la definición de una red bayesiana, en este caso constituyendo su parte cualitativa.

- **bloque temático 3**: inferencia en redes bayesianas a partir de una serie de hallazgos que se introducen en la red.

Cada uno de estos bloques temáticos darán lugar a distintos tipos de preguntas incorporando una parametrización que hace posible la creación de distintas instancias o variaciones sobre el mismo tipo de pregunta definido para dar lugar a las preguntas finales que formaran parte de un examen o test de autoevaluación y la asignatura correspondiente en *Siette*. En la tabla 3.3 se identifican las tareas de análisis mencionadas.

Tabla 3.3: Tareas de Análisis.

TAREAS DE ANÁLISIS	DESCRIPCIÓN
LGAP_ANA_COMUN	Parte común a todos los tipos de preguntas
LGAP_ANA_RELIPDP	Bloque temático 1
LGAP_ANA_RELIPGRAFOS	Bloque temático 2
LGAP_ANA_RB	Bloque temático 3

En la figura 3.1 se presenta el detalle en la planificación de las tareas de análisis:

Figura 3.1: Detalle de las tareas de Análisis en el diagrama de Gantt.

3.2.2. Tareas de Diseño

La nomenclatura utilizada para este tipo de tareas será la siguiente:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.
- palabra ***DIS*** que es la abreviatura de la palabra diseño para indicar que se trata de una tarea de diseño.
- palabra significativa que describe la acción que se lleva acabo con la ejecución de cada tarea.

Partiendo de la tarea de análisis ***LGAP_ANA_COMUN*** definida en el punto anterior llegamos a la conclusión que será necesario tener una serie de componentes comunes a cualquier tipo de pregunta que se quiera definir en la librería. Los componentes a definir, que dan lugar a cada una de las tareas de diseño, son los siguientes:

- *LGAP_DIS_EXCEPCIONES* Se definirán en el sistema un manejo de excepciones propio para identificar las posibles circunstancias de error o *warnings* que se pueden producir en la ejecución de la librería.

- *LGAP_DIS_UTIL* este componente reunirá todas las funcionalidades comunes al resto de componentes como puede ser la generación aleatoria de números reales, algoritmos comunes para la generación de distribuciones de probabilidad, formateo de la salida, etc.

- *LGAP_DIS_EXAMEN* definirá la base para la generación de exámenes de test entendidos como una colección de preguntas obtenidas al azar entre el repertorio ofrecido por la librería de generación automática de preguntas.

- *LGAP_DIS_SIETTE* establecerá cuál será la interfaz necesaria para la integración de la librería desarrollada dentro del sistema *Siette*. En este punto se diseña la interfaz que debe implementar una clase para su integración en el sistema *Siette*.

El resto de tareas de diseño son las correspondientes a la definición de cada tipo de pregunta que se va a plantear y que se analizaron en el apartado anterior. [4]Identificamos las siguientes tareas de diseño:

- *LGAP_DIS_RELIPDP* Para los tipos de preguntas de la 1 a la 4 incluida, definiremos un sistema compuesto por un conjunto de variables aleatorias y una distribución de probabilidad definida sobre las distintas combinaciones de valores que pueden tomar cada una de las variables aleatorias que lo componen. Esta será la base para el planteamiento de relaciones de independencia y de independencia probabilística condicional.

- *LGAP_DIS_GRAFOS* Para los tipos de pregunta 5 y 6 definiremos las clases que permitirán modelar grafos acíclicos sobre los que también se plantearán relaciones de independencia y de independencia probabilística condicional, esta vez sobre los nodos de un grafo que representan igualmente a variables aleatorias. Las preguntas de tipo 5 serán relativas a grafos no dirigidos acíclicos (GNA) y las de tipo 6 corresponderán a grafos dirigidos acíclicos (GDA).

- *LGAP_DIS_RB* Por último para los tipos de pregunta 7, 8 y 9, plantearemos distintos tipos de pregunta que tienen que ver con la inferencia en redes bayesianas y enmarcadas en el tercer bloque temático. Se planteará una red bayesiana sobre la que se introducen uno o varios hallazgos sobre una o varias variables o nodos de la red para posteriormente preguntar por algún valor predictivo de la variable de interés, tras la evidencia introducida en la red.

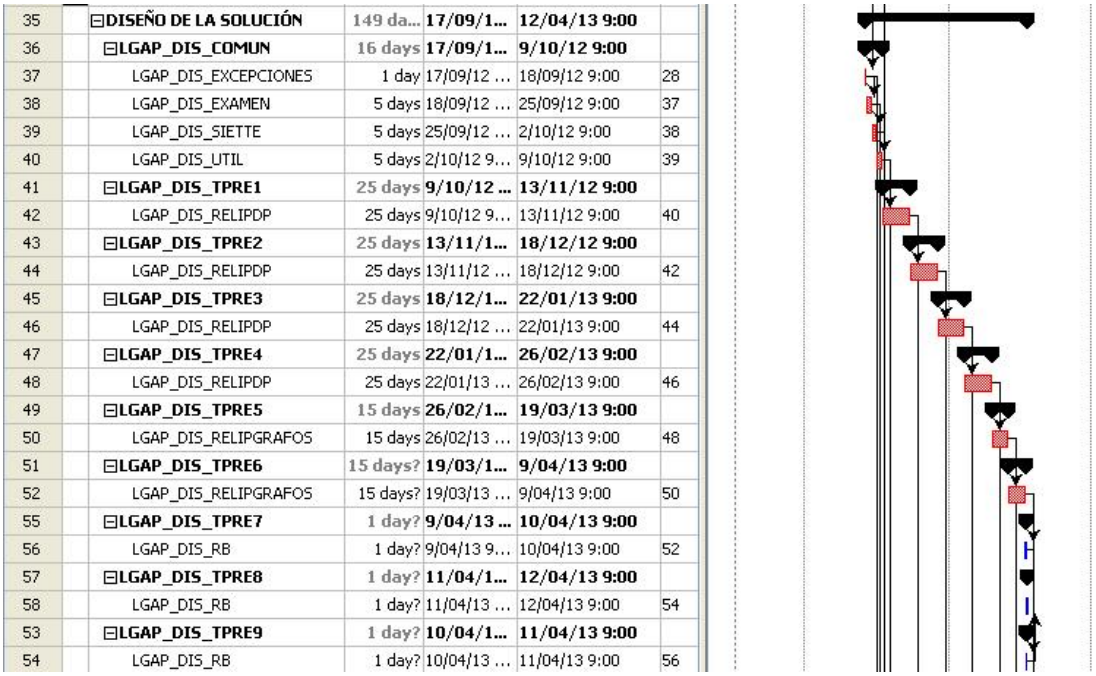
[4]Nombraremos como *LGAP_DIS_TPER?* cada uno de los tipos de preguntas, donde el carácter '?' será sustituido por el correspondiente número de orden.

En la tabla 3.4 se muestran las distintas tareas de diseño identificadas en este apartado:

Tabla 3.4: Tareas de Diseño.

TAREA DE DISEÑO	**RESULTADO**
LGAP_DIS_EXCEPCIONES	Diseño del manejador de excepciones
LGAP_DIS_UTIL	Diseño del componente de utilidades
LGAP_DIS_EXAMEN	Confección del examen PDF
LGAP_DIS_SIETTE	Diseño del interfaz con *Siette*
LGAP_DIS_TPRE1	Sistema definido en LGAP_DIS_RELIPDP
LGAP_DIS_TPRE2	Sistema definido en LGAP_DIS_RELIPDP
LGAP_DIS_TPRE3	Sistema definido en LGAP_DIS_RELIPDP
LGAP_DIS_TPRE4	Sistema definido en LGAP_DIS_RELIPDP
LGAP_DIS_TPRE5	Sistema definido en LGAP_DIS_RELIPGRAFOS
LGAP_DIS_TPRE6	Sistema definido en LGAP_DIS_RELIPGRAFOS
LGAP_DIS_TPRE7	Sistema definido en LGAP_DIS_RB
LGAP_DIS_TPRE8	Sistema definido en LGAP_DIS_RB
LGAP_DIS_TPRE9	Sistema definido en LGAP_DIS_RB

En la figura 3.2 se presenta el *zoom* sobre las tareas de diseño del diagrama de Gantt de la planificación del proyecto:



35	**DISEÑO DE LA SOLUCIÓN**	149 da...	**17/09/1...**	**12/04/13 9:00**	
36	**LGAP_DIS_COMUN**	16 days	**17/09/1...**	**9/10/12 9:00**	
37	LGAP_DIS_EXCEPCIONES	1 day	17/09/12 ...	18/09/12 9:00	28
38	LGAP_DIS_EXAMEN	5 days	18/09/12 ...	25/09/12 9:00	37
39	LGAP_DIS_SIETTE	5 days	25/09/12 ...	2/10/12 9:00	38
40	LGAP_DIS_UTIL	5 days	2/10/12 9...	9/10/12 9:00	39
41	**LGAP_DIS_TPRE1**	25 days	**9/10/12 ...**	**13/11/12 9:00**	
42	LGAP_DIS_RELIPDP	25 days	9/10/12 9...	13/11/12 9:00	40
43	**LGAP_DIS_TPRE2**	25 days	**13/11/1...**	**18/12/12 9:00**	
44	LGAP_DIS_RELIPDP	25 days	13/11/12 ...	18/12/12 9:00	42
45	**LGAP_DIS_TPRE3**	25 days	**18/12/1...**	**22/01/13 9:00**	
46	LGAP_DIS_RELIPDP	25 days	18/12/12 ...	22/01/13 9:00	44
47	**LGAP_DIS_TPRE4**	25 days	**22/01/1...**	**26/02/13 9:00**	
48	LGAP_DIS_RELIPDP	25 days	22/01/13 ...	26/02/13 9:00	46
49	**LGAP_DIS_TPRE5**	15 days	**26/02/1...**	**19/03/13 9:00**	
50	LGAP_DIS_RELIPGRAFOS	15 days	26/02/13 ...	19/03/13 9:00	48
51	**LGAP_DIS_TPRE6**	15 days?	**19/03/1...**	**9/04/13 9:00**	
52	LGAP_DIS_RELIPGRAFOS	15 days?	19/03/13 ...	9/04/13 9:00	50
55	**LGAP_DIS_TPRE7**	1 day?	**9/04/13 ...**	**10/04/13 9:00**	
56	LGAP_DIS_RB	1 day?	9/04/13 9...	10/04/13 9:00	52
57	**LGAP_DIS_TPRE8**	1 day?	**11/04/1...**	**12/04/13 9:00**	
58	LGAP_DIS_RB	1 day?	11/04/13 ...	12/04/13 9:00	54
53	**LGAP_DIS_TPRE9**	1 day?	**10/04/1...**	**11/04/13 9:00**	
54	LGAP_DIS_RB	1 day?	10/04/13 ...	11/04/13 9:00	56

Figura 3.2: Detalle de las tareas de Diseño en el diagrama de Gantt.

3.2.3. Tareas de Implementación

La nomenclatura que se va a utilizar para definir las tareas de implementación queda definida de la siguiente manera:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.
- palabra ***IMP*** que es la abreviatura de la palabra implementación para indicar que se trata de una tarea de implementación.
- palabra que describe la acción concreta que se lleva acabo con la ejecución de cada tarea.

Cada tarea de diseño identificada en el apartado anterior, tendrá su correspondiente tarea de implementación, la correspondencia será de una a una. Esto es así ya que en teoría el paso del diseño a la codificación debería ser un proceso automático o semiautomático. El diseño realizado, independientemente del lenguaje de programación, responde sin embargo a un diseño *OO* y en este punto del proyecto pasamos a su implementación final en el lenguaje de programación Java.

Así podemos identificar las siguientes tareas de implementación. Ver tabla 3.5:

Tabla 3.5: Tareas de Implementación.

TAREA DE IMPLEMENTACIÓN	**RESULTADO**
LGAP_IMP_EXCEPCIONES	Manejador de excepciones
LGAP_IMP_UTIL	Componente de utilidades
LGAP_IMP_EXAMEN	Examen PDF
LGAP_IMP_SIETTE	Interfaz con *Siette*
LGAP_IMP_TPRE1	Sistema definido en LGAP_IMP_RELIPDP
LGAP_IMP_TPRE2	Sistema definido en LGAP_IMP_RELIPDP
LGAP_IMP_TPRE3	Sistema definido en LGAP_IMP_RELIPDP
LGAP_IMP_TPRE4	Sistema definido en LGAP_IMP_RELIPDP
LGAP_IMP_TPRE5	Sistema definido en LGAP_IMP_RELIPGRAFOSGNA
LGAP_IMP_TPRE6	Sistema definido en LGAP_IMP_RELIPGRAFOSGDA
LGAP_IMP_TPRE7	Sistema definido en LGAP_IMP_RB
LGAP_IMP_TPRE8	Sistema definido en LGAP_IMP_RB
LGAP_IMP_TPRE9	Sistema definido en LGAP_IMP_RB

En la figura 3.3 representamos las tareas identificadas de implementación con el correspondiente zoom del diagrama de Gantt:

59	⊟IMPLEMENTACIÓN	151 da...	1/10/12 ...	30/04/13 9:00	31
60	⊟LGAP_IMP_COMUN	11 days	1/10/12 ...	16/10/12 9:00	
61	LGAP_IMP_EXCEPCIONES	1 day	1/10/12 9...	2/10/12 9:00	37
62	LGAP_IMP_RELIP	10 days	1/10/12 9...	15/10/12 9:00	38
63	LGAP_IMP_SIETTE	10 days	2/10/12 9...	16/10/12 9:00	39
64	LGAP_IMP_UTIL	5 days	9/10/12 9...	16/10/12 9:00	40
65	⊟LGAP_IMP_TPRE1	25 days	13/11/1...	18/12/12 9:00	
66	LGAP_IMP_SISTEMA	25 days	13/11/12 ...	18/12/12 9:00	42
67	⊟LGAP_IMP_TPRE2	10 days	18/12/1...	1/01/13 9:00	
68	LGAP_IMP_SISTEMA	10 days	18/12/12 ...	1/01/13 9:00	44
69	⊟LGAP_IMP_TPRE3	10 days	22/01/1...	5/02/13 9:00	
70	LGAP_IMP_SISTEMA	10 days	22/01/13 ...	5/02/13 9:00	46
71	⊟LGAP_IMP_TPRE4	10 days	26/02/1...	12/03/13 9:00	
72	LGAP_IMP_SISTEMA	10 days	26/02/13 ...	12/03/13 9:00	48
73	⊟LGAP_IMP_TPRE5	15 days	19/03/1...	9/04/13 9:00	
74	LGAP_IMP_GRAFOSGNA	15 days	19/03/13 ...	9/04/13 9:00	50
75	⊟LGAP_IMP_TPRE6	15 days	9/04/13 ...	30/04/13 9:00	
76	LGAP_IMP_GRAFOSGDA	15 days	9/04/13 9...	30/04/13 9:00	52
77	⊟LGAP_IMP_TPRE7	1 day?	10/04/1...	11/04/13 9:00	
78	LGAP_IMP_RB	1 day?	10/04/13 ...	11/04/13 9:00	54
79	⊟LGAP_IMP_TPRE8	1 day?	12/04/1...	15/04/13 9:00	
80	LGAP_IMP_RB	1 day?	12/04/13 ...	15/04/13 9:00	56
81	⊟LGAP_IMP_TPRE9	1 day?	11/04/1...	12/04/13 9:00	
82	LGAP_IMP_RB	1 day?	11/04/13 ...	12/04/13 9:00	58

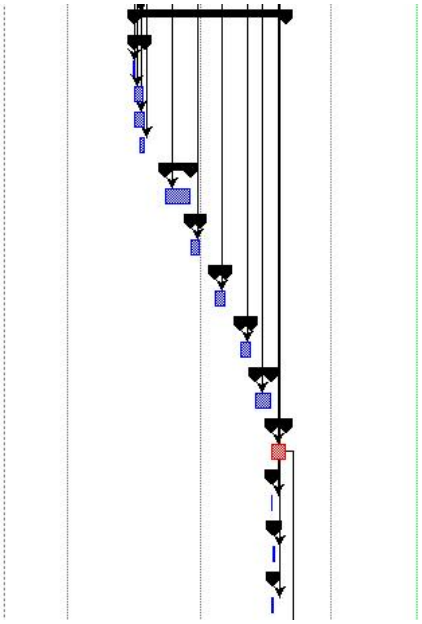


Figura 3.3: Detalle de las tareas de Implementación en el diagrama de Gantt.

3.2.4. Tareas de Pruebas unitarias y eficiencia

La nomenclatura que se va a utilizar para el caso de las tareas de pruebas unitarias y de eficiencia será la siguiente:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.
- palabra ***PRU*** que es la abreviatura de prueba para indicar que se trata de una tarea de prueba unitaria.
- palabra que describe la acción que se lleva acabo con la ejecución de cada tarea identificando el componente sobre el que se realiza la prueba unitaria o de eficiencia.

Haciendo uso del *framework* JUnit incorporado en el entorno de desarrollo Eclipse, se definirán una serie de casos de prueba en los siguientes apartados:

- *LGAP_PRU_TPRE?* Cada tipo de pregunta definida tendrá su correspondiente caso de prueba. Se definirá un test en JUnit para cada una de los distintos tipos de pregunta que se van a plantear a lo largo del proyecto. Cada uno de los nueve tipos de pregunta tiene una parametrización propia que nos permitirá ir obteniendo cada una de la distintas preguntas finales que genera la librería y que formaran parte de la asignatura que se defina finalmente en el sistema *Siette*. Además dichos

test tomarán la medida del tiempo que tarda en ejecutarse la generación de cada pregunta para obtener una serie de estadísticas que nos permitan realizar un estudio sobre la eficiencia del sistema desarrollado.

- *LGAP_PRU_GNA* y *LGAP_PRU_GDA* Un apartado importante son las preguntas relacionadas con grafos. Para realizar pruebas de grano fino sobre grafos, se realizarán test específicos para la definición y representación gráfica de los grafos. Tendremos pues un test para realizar las pruebas sobre GNA que permitirán probar los algoritmos que calculan los caminos activos entre dos nodos cualesquiera de un grafo no dirigido acíclico. De manera análoga tendremos un test sobre GDA para realizar las mismas pruebas ya comentadas en los GNA esta vez sobre grafos dirigidos acíclicos. La representación gráfica del grafo tendrá un papel fundamental en la resolución de cada pregunta y a la hora de obtener el refuerzo asociado a cada tipo de pregunta planteada.

En la tabla 3.6 se identifican las tareas correspondientes a las pruebas unitarias realizadas sobre cada componente de la librería.

Tabla 3.6: Pruebas Unitarias.

PRUEBAS UNITARIAS	RESULTADO
LGAP_PRU_TPRE1	Plan de prueba para el tipo de pregunta 1
LGAP_PRU_TPRE2	Plan de prueba para el tipo de pregunta 2
LGAP_PRU_TPRE3	Plan de prueba para el tipo de pregunta 3
LGAP_PRU_TPRE4	Plan de prueba para el tipo de pregunta 4
LGAP_PRU_GNA	Plan de prueba para grafos GNA
LGAP_PRU_GDA	Plan de prueba para grafos GDA
LGAP_PRU_TPRE5	Plan de prueba para el tipo de pregunta 5
LGAP_PRU_TPRE6	Plan de prueba para el tipo de pregunta 6
LGAP_PRU_TPRE7	Plan de prueba para el tipo de pregunta 7
LGAP_PRU_TPRE8	Plan de prueba para el tipo de pregunta 8
LGAP_PRU_TPRE9	Plan de prueba para el tipo de pregunta 9

En la figura 3.4 se muestra el detalle de las tareas correspondientes a las pruebas unitarias. Como se ha comentado anteriormente, en estas tareas se consideran también las pruebas de eficiencia en la generación de cada tipo de pregunta, registrando los tiempos empleados en la generación de cada una de ellas para su posterior estudio.

83	⊟PRUEBAS UNITARIAS	27,5 da...	30/04/1...	6/06/13 14:00	
84	LGAP_PRU_TPRE1	2,5 days	30/04/13 ...	2/05/13 14:00	76
85	LGAP_PRU_TPRE2	2,5 days	2/05/13 1...	7/05/13 9:00	84
86	LGAP_PRU_TPRE3	2,5 days	7/05/13 9...	9/05/13 14:00	85
87	LGAP_PRU_TPRE4	2,5 days	9/05/13 1...	14/05/13 9:00	86
88	LGAP_PRU_TPRE5	2,5 days	14/05/13 ...	16/05/13 14:00	87
89	LGAP_PRU_TPRE6	2,5 days	16/05/13 ...	21/05/13 9:00	88
90	LGAP_PRU_GNA	2,5 days	21/05/13 ...	23/05/13 14:00	89
91	LGAP_PRU_GDA	2,5 days	23/05/13 ...	28/05/13 9:00	90
92	LGAP_PRU_TPRE7	2,5 days	28/05/13 ...	30/05/13 14:00	91
93	LGAP_PRU_TPRE8	2,5 days	30/05/13 ...	4/06/13 9:00	92
94	LGAP_PRU_TPRE9	2,5 days	4/06/13 9...	6/06/13 14:00	93

Figura 3.4: Detalle de las Pruebas Unitarias en el diagrama de Gantt.

3.2.5. Tareas de Documentación

En este apartado se van a identificar las tareas que tienen que ver con la documentación del proyecto de fin de carrera realizado así como de toda la documentación técnica de la librería desarrollada y de los manuales de usuario.

La nomenclatura que se va a utilizar será la siguiente:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.

- palabra ***DOC*** que es la abreviatura de la palabra documentación.

- palabra que describe la acción que se lleva acabo con la ejecución de cada tarea identificando el componente sobre el que se escribe la documentación.

Tendremos las siguientes tareas:

LGAP_DOC_MEMORIA es la tarea de redacción de la memoria presente del proyecto de fin de carrera.

LGAP_DOC_MANUALES se crearan los manuales de usuario correspondiente al uso de la librería desde un programa cliente y su integración en el sistema *Siette* (ver apéndices **??**). Además se dan las directrices que debe seguir un desarrollador o estudiante de ingeniería que quiera contribuir a seguir desarrollando el proyecto en la línea de incorporar nuevos tipos de preguntas de la asignatura propuesta en este proyecto o de otras.

La tabla 3.7 muestra las tareas de documentación identificadas:

Tabla 3.7: Tareas de Documentación.

TAREAS DE DOCUMENTACIÓN	RESULTADO
LGAP_DOC_MEMORIA	Tarea de confección y redacción de la memoria del proyecto
LGAP_DOC_MANUALES_USU	Redacción de los manuales de usuario

A continuación 3.5 presentamos el detalle, sobre el diagrama de Gantt, de las tareas de documentación:

Figura 3.5: Detalle de las tareas de Documentación en el diagrama de Gantt.

3.2.6. Tareas de Implantación y despliegue

La etapa final en el desarrollo del proyecto será su implantación para que sea utilizado desde un programa cliente Java. Además, como requisito fundamental de este proyecto, la librería desarrollada ha de poder ser integrada y desplegada en el sistema *Siette* para ser el motor de generación de preguntas de la asignatura definida sobre Modelos Gráficos Probabilistas y que se abastece de las preguntas que se generan de forma automática por la librería desarrollada.

La nomenclatura en este caso en la siguiente:

- palabra ***LGAP*** que proviene del nombre abreviado que se le va a dar a este proyecto: **L**ibrería de **G**eneración **A**utomática de **P**reguntas.
- palabra ***IMPL*** que es la abreviatura de la palabra implantación.
- palabra que describe la acción que se lleva acabo con la implantación o despliegue de la librería generada.

Identificamos a continuación cada una de las tareas en esta fase del ciclo de vida proyecto:

- *LGAP_IMPL_PROG* implantación para ser utilizado desde un programa cliente. Se desarrollará un sencillo interfaz de usuario haciendo uso de la tecnología JSP que nos permitirá generar exámenes de test en formato PDF haciendo uso de la librería desarrollada. El examen se compondrá de un número determinado de preguntas de

test en las que se plantea el enunciado de una pregunta y las posibles respuestas a la pregunta planteada. El interfaz tendrá que ofrecer la posibilidad de generar el examen planteado y el mismo marcando las respuestas correctas a cada pregunta con la información de refuerzo correspondiente. Además se podrá elegir el número de preguntas de las que se compone el examen así como el número de respuestas o alternativas que se presentarán como solución posible a la pregunta planteada.

- *LGAP_IMPL_SIETTE* tarea correspondiente a la integración de la librería desarrollada en el sistema *Siette*. Se creará una nueva asignatura en dicho sistema y se utiliza la librería para ir definiendo cada una de las preguntas de dicha asignatura. Cada tipo de pregunta ofrecida por la librería se instanciará de manera que tengamos un repertorio lo suficientemente amplio como para poder generar posteriormente exámenes o test de autoevaluación completos. Las instancias que se definiran en *Siette* estarán en torno a las treinta o cuarenta preguntas que serán utilizadas para la elaboración de los test de autoevaluación.

En la tabla 3.8 identificamos cada una de las tareas correspondientes a la implantación del trabajo desarrollado.

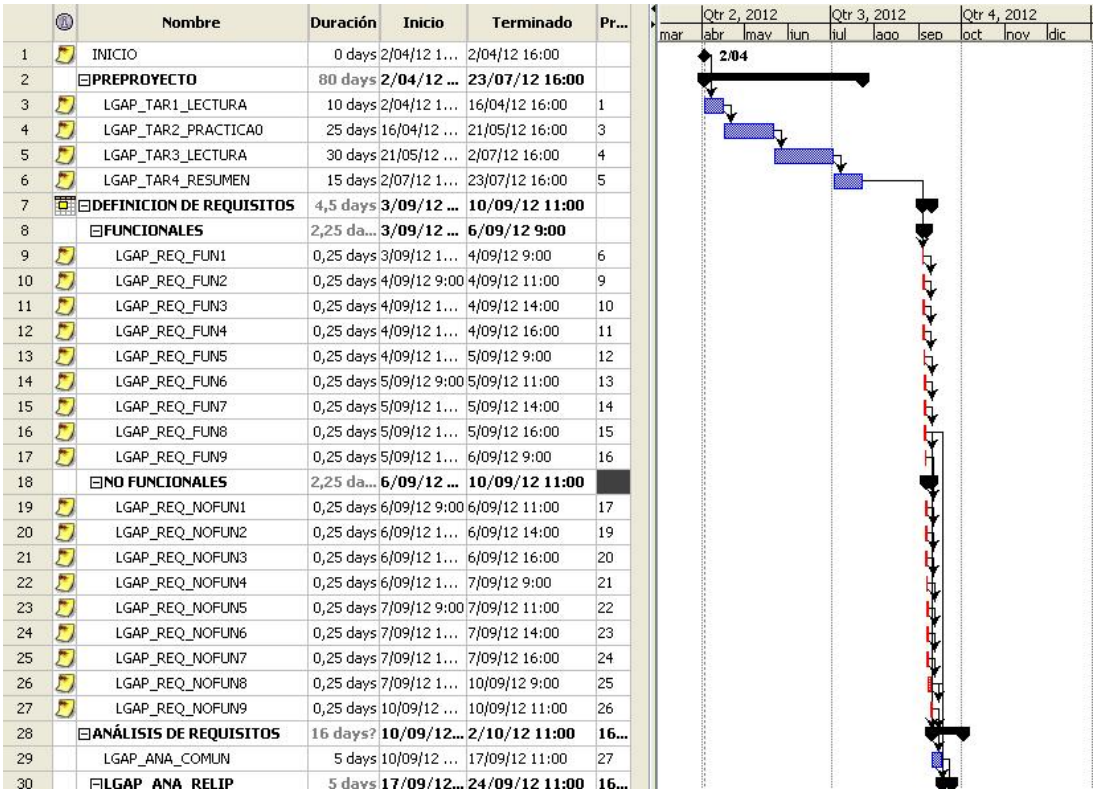
Tabla 3.8: Tareas de Implantación y despliegue.

TAREAS DE IMPLANTACIÓN	RESULTADO
LGAP_IMPL_PROG	Tarea de implantación para el uso desde un programa cliente
LGAP_IMPL_SIETTE	Tarea de implantación para la integración con *Siette*

Por último en la figura 3.6 se presenta el detalle correspondiente a las tareas de implantación.

Figura 3.6: Detalle de las tareas de Implantación en el diagrama de Gantt.

3.3. Planificación del proyecto sobre un Diagrama de Gantt

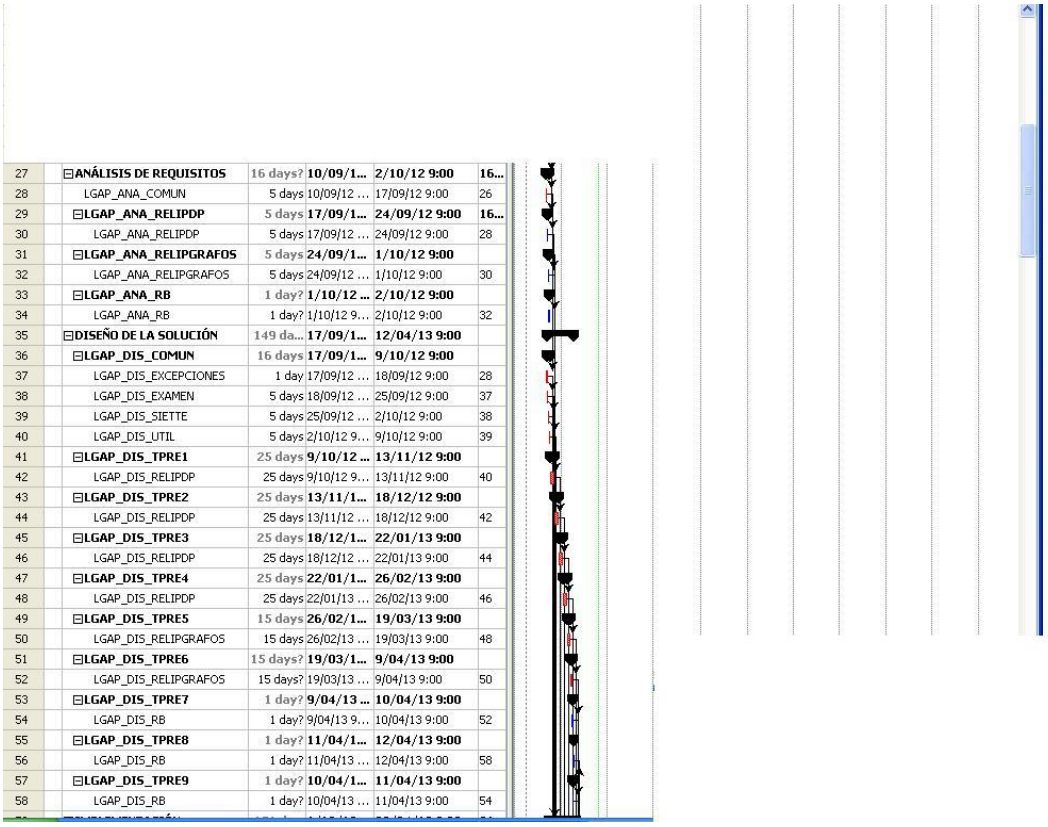


	Nombre	Duración	Inicio	Terminado	Pr...
1	INICIO	0 days	2/04/12 1...	2/04/12 16:00	
2	**PREPROYECTO**	80 days	**2/04/12 ...**	**23/07/12 16:00**	
3	LGAP_TAR1_LECTURA	10 days	2/04/12 1...	16/04/12 16:00	1
4	LGAP_TAR2_PRACTICA0	25 days	16/04/12 ...	21/05/12 16:00	3
5	LGAP_TAR3_LECTURA	30 days	21/05/12 ...	2/07/12 16:00	4
6	LGAP_TAR4_RESUMEN	15 days	2/07/12 1...	23/07/12 16:00	5
7	**DEFINICION DE REQUISITOS**	4,5 days	**3/09/12 ...**	**10/09/12 11:00**	
8	**FUNCIONALES**	2,25 da...	**3/09/12 ...**	**6/09/12 9:00**	
9	LGAP_REQ_FUN1	0,25 days	3/09/12 1...	4/09/12 9:00	6
10	LGAP_REQ_FUN2	0,25 days	4/09/12 9:00	4/09/12 11:00	9
11	LGAP_REQ_FUN3	0,25 days	4/09/12 1...	4/09/12 14:00	10
12	LGAP_REQ_FUN4	0,25 days	4/09/12 1...	4/09/12 16:00	11
13	LGAP_REQ_FUN5	0,25 days	4/09/12 1...	5/09/12 9:00	12
14	LGAP_REQ_FUN6	0,25 days	5/09/12 9:00	5/09/12 11:00	13
15	LGAP_REQ_FUN7	0,25 days	5/09/12 1...	5/09/12 14:00	14
16	LGAP_REQ_FUN8	0,25 days	5/09/12 1...	5/09/12 16:00	15
17	LGAP_REQ_FUN9	0,25 days	5/09/12 1...	6/09/12 9:00	16
18	**NO FUNCIONALES**	2,25 da...	**6/09/12 ...**	**10/09/12 11:00**	
19	LGAP_REQ_NOFUN1	0,25 days	6/09/12 9:00	6/09/12 11:00	17
20	LGAP_REQ_NOFUN2	0,25 days	6/09/12 1...	6/09/12 14:00	19
21	LGAP_REQ_NOFUN3	0,25 days	6/09/12 1...	6/09/12 16:00	20
22	LGAP_REQ_NOFUN4	0,25 days	6/09/12 1...	7/09/12 9:00	21
23	LGAP_REQ_NOFUN5	0,25 days	7/09/12 9:00	7/09/12 11:00	22
24	LGAP_REQ_NOFUN6	0,25 days	7/09/12 1...	7/09/12 14:00	23
25	LGAP_REQ_NOFUN7	0,25 days	7/09/12 1...	7/09/12 16:00	24
26	LGAP_REQ_NOFUN8	0,25 days	7/09/12 1...	10/09/12 9:00	25
27	LGAP_REQ_NOFUN9	0,25 days	10/09/12 ...	10/09/12 11:00	26
28	**ANÁLISIS DE REQUISITOS**	16 days?	**10/09/12...**	**2/10/12 11:00**	16...
29	LGAP_ANA_COMUN	5 days	10/09/12 ...	17/09/12 11:00	27
30	**LGAP_ANA_RELIP**	5 days	**17/09/12...**	**24/09/12 11:00**	16...

Figura 3.7: Diagrama de Gantt completo del proyecto (1/4).

En este apartado vamos a realizar una planificación del proyecto de desarrollo de software a realizar identificando cada acción a realizar con una tarea de manera que pueda considerarse como un bloque unitario de trabajo bien definido y que puede asignarse a un determinado recurso para su realización con una fecha de inicio de la tarea, una duración de la misma y una fecha de finalización. El diagrama de Gantt nos permite ver como se van desarrollando estas tareas a lo largo del tiempo y las dependencias que se establecen entre ellas en el sentido de que una determinada tarea no pueda dar comienzo hasta que otras tareas dependientes hayan sido realizadas.

A partir de la identificación de tareas realizadas en los apartados anteriores, en las siguientes figuras se representa el diagrama de Gantt completo que permite ver la planificación final del proyecto a lo largo del tiempo. Observar que en el diagrama no se ha realizado la asignación de recursos ya que se trata de un proyecto individual y el recurso al que se asignan las tareas es siempre el mismo, el alumno que realiza este proyecto de



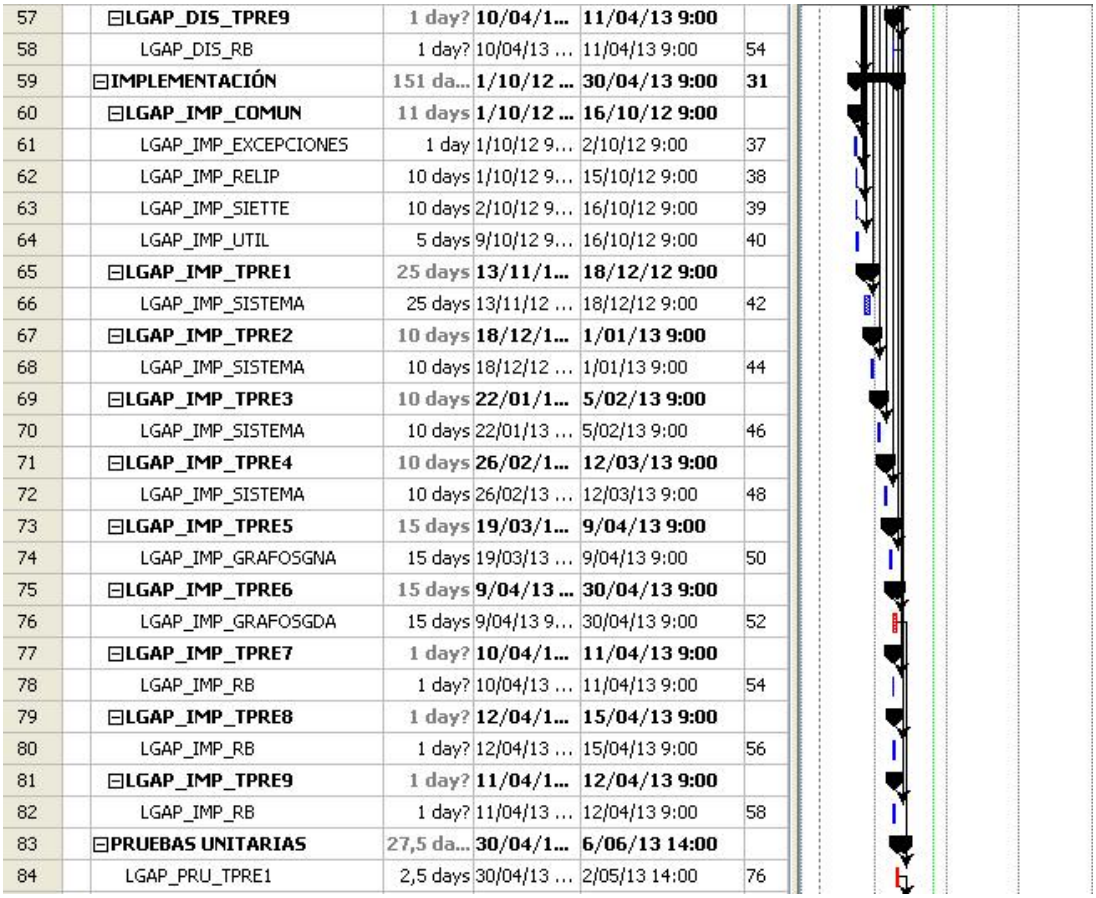
27	⊟ANÁLISIS DE REQUISITOS	16 days?	10/09/1...	2/10/12 9:00	16...
28	LGAP_ANA_COMUN	5 days	10/09/12 ...	17/09/12 9:00	26
29	⊟LGAP_ANA_RELIPDP	5 days	17/09/1...	24/09/12 9:00	16...
30	LGAP_ANA_RELIPDP	5 days	17/09/12 ...	24/09/12 9:00	28
31	⊟LGAP_ANA_RELIPGRAFOS	5 days	24/09/1...	1/10/12 9:00	
32	LGAP_ANA_RELIPGRAFOS	5 days	24/09/12 ...	1/10/12 9:00	30
33	⊟LGAP_ANA_RB	1 day?	1/10/12 ...	2/10/12 9:00	
34	LGAP_ANA_RB	1 day?	1/10/12 9...	2/10/12 9:00	32
35	⊟DISEÑO DE LA SOLUCIÓN	149 da...	17/09/1...	12/04/13 9:00	
36	⊟LGAP_DIS_COMUN	16 days	17/09/1...	9/10/12 9:00	
37	LGAP_DIS_EXCEPCIONES	1 day	17/09/12 ...	18/09/12 9:00	28
38	LGAP_DIS_EXAMEN	5 days	18/09/12 ...	25/09/12 9:00	37
39	LGAP_DIS_SIETTE	5 days	25/09/12 ...	2/10/12 9:00	38
40	LGAP_DIS_UTIL	5 days	2/10/12 9...	9/10/12 9:00	39
41	⊟LGAP_DIS_TPRE1	25 days	9/10/12 ...	13/11/12 9:00	
42	LGAP_DIS_RELIPDP	25 days	9/10/12 9...	13/11/12 9:00	40
43	⊟LGAP_DIS_TPRE2	25 days	13/11/1...	18/12/12 9:00	
44	LGAP_DIS_RELIPDP	25 days	13/11/12 ...	18/12/12 9:00	42
45	⊟LGAP_DIS_TPRE3	25 days	18/12/1...	22/01/13 9:00	
46	LGAP_DIS_RELIPDP	25 days	18/12/12 ...	22/01/13 9:00	44
47	⊟LGAP_DIS_TPRE4	25 days	22/01/1...	26/02/13 9:00	
48	LGAP_DIS_RELIPDP	25 days	22/01/13 ...	26/02/13 9:00	46
49	⊟LGAP_DIS_TPRE5	15 days	26/02/1...	19/03/13 9:00	
50	LGAP_DIS_RELIPGRAFOS	15 days	26/02/13 ...	19/03/13 9:00	48
51	⊟LGAP_DIS_TPRE6	15 days?	19/03/1...	9/04/13 9:00	
52	LGAP_DIS_RELIPGRAFOS	15 days?	19/03/13 ...	9/04/13 9:00	50
53	⊟LGAP_DIS_TPRE7	1 day?	9/04/13 ...	10/04/13 9:00	
54	LGAP_DIS_RB	1 day?	9/04/13 9...	10/04/13 9:00	52
55	⊟LGAP_DIS_TPRE8	1 day?	11/04/1...	12/04/13 9:00	
56	LGAP_DIS_RB	1 day?	11/04/13 ...	12/04/13 9:00	58
57	⊟LGAP_DIS_TPRE9	1 day?	10/04/1...	11/04/13 9:00	
58	LGAP_DIS_RB	1 day?	10/04/13 ...	11/04/13 9:00	54

Figura 3.8: Diagrama de Gantt completo del proyecto (2/4).

fin de carrera.

También se ha considerado en este diagrama final, las tareas previas que se llevaron a cabo como anteproyecto y que corresponden a la investigación de campo y el estudio de la base teórica sobre la que están basadas cada uno de los distintos bloques temáticos o tipos de preguntas que ofrece la librería desarrollada, así como la realización de la llamada práctica cero propuesta por el equipo docente para familiarizarse con las distintas herramientas software de las que se hace uso para la realización del este proyecto de fin de carrera.

Asimismo se han incluido también en el diagrama las actividades posteriores, aquellas que se realizan una vez finalizadas las tareas de documentación, implantación y despliegue del software completamente desarrollado. Estas actividades pueden ser por ejemplo, la creación de la página web o wikipedia del proyecto donde reside toda la información relacionada con este proyecto de cara a futuras ampliaciones del mismo o colaboraciones de terceros como proyecto de software libre que se trata. Esta wiki del proyecto se puede visitar para su consulta en la siguiente dirección de Internet: Wiki. Otra actividad que se refleja en el diagrama de planificación del proyecto es la elaboración y preparación de la



57	⊟**LGAP_DIS_TPRE9**	1 day?	**10/04/1...**	**11/04/13 9:00**	
58	LGAP_DIS_RB	1 day?	10/04/13 ...	11/04/13 9:00	54
59	⊟**IMPLEMENTACIÓN**	151 da...	**1/10/12 ...**	**30/04/13 9:00**	**31**
60	⊟**LGAP_IMP_COMUN**	11 days	**1/10/12 ...**	**16/10/12 9:00**	
61	LGAP_IMP_EXCEPCIONES	1 day	1/10/12 9...	2/10/12 9:00	37
62	LGAP_IMP_RELIP	10 days	1/10/12 9...	15/10/12 9:00	38
63	LGAP_IMP_SIETTE	10 days	2/10/12 9...	16/10/12 9:00	39
64	LGAP_IMP_UTIL	5 days	9/10/12 9...	16/10/12 9:00	40
65	⊟**LGAP_IMP_TPRE1**	25 days	**13/11/1...**	**18/12/12 9:00**	
66	LGAP_IMP_SISTEMA	25 days	13/11/12 ...	18/12/12 9:00	42
67	⊟**LGAP_IMP_TPRE2**	10 days	**18/12/1...**	**1/01/13 9:00**	
68	LGAP_IMP_SISTEMA	10 days	18/12/12 ...	1/01/13 9:00	44
69	⊟**LGAP_IMP_TPRE3**	10 days	**22/01/1...**	**5/02/13 9:00**	
70	LGAP_IMP_SISTEMA	10 days	22/01/13 ...	5/02/13 9:00	46
71	⊟**LGAP_IMP_TPRE4**	10 days	**26/02/1...**	**12/03/13 9:00**	
72	LGAP_IMP_SISTEMA	10 days	26/02/13 ...	12/03/13 9:00	48
73	⊟**LGAP_IMP_TPRE5**	15 days	**19/03/1...**	**9/04/13 9:00**	
74	LGAP_IMP_GRAFOSGNA	15 days	19/03/13 ...	9/04/13 9:00	50
75	⊟**LGAP_IMP_TPRE6**	15 days	**9/04/13 ...**	**30/04/13 9:00**	
76	LGAP_IMP_GRAFOSGDA	15 days	9/04/13 9...	30/04/13 9:00	52
77	⊟**LGAP_IMP_TPRE7**	1 day?	**10/04/1...**	**11/04/13 9:00**	
78	LGAP_IMP_RB	1 day?	10/04/13 ...	11/04/13 9:00	54
79	⊟**LGAP_IMP_TPRE8**	1 day?	**12/04/1...**	**15/04/13 9:00**	
80	LGAP_IMP_RB	1 day?	12/04/13 ...	15/04/13 9:00	56
81	⊟**LGAP_IMP_TPRE9**	1 day?	**11/04/1...**	**12/04/13 9:00**	
82	LGAP_IMP_RB	1 day?	11/04/13 ...	12/04/13 9:00	58
83	⊟**PRUEBAS UNITARIAS**	27,5 da...	**30/04/1...**	**6/06/13 14:00**	
84	LGAP_PRU_TPRE1	2,5 days	30/04/13 ...	2/05/13 14:00	76

Figura 3.9: Diagrama de Gantt completo del proyecto (3/4).

presentación utilizada para la defensa ante el tribunal del proyecto de fin de carrera.

En las figuras 3.7, 3.8, 3.9, 3.10 representamos el diagrama de Gantt completo que presenta la planificación total del proyecto en todas las etapas de su ciclo de vida (definición, análisis, diseño, implementación, pruebas y documentación), donde podemos ver todas estas actividades y tareas comentadas en los párrafos anteriores.

3.4. Estudio de Viabilidad. Estudio del plan de proyecto

A partir del estudio del plan del proyecto detallado en el apartado anterior, se determinará la viabilidad de afrontarlo como proyecto de fin de carrera. En el sistema actual, el crédito representa el número de horas de clase que un profesor imparte. En concreto, un crédito actual corresponde a 10 horas lectivas (o 10 horas "de clase"). El crédito europeo, sin embargo, mide el volumen o carga total del trabajo de aprendizaje del estudiante

82		LGAP_IMP_RB	1 day?	11/04/13 ...	12/04/13 9:00	58
83		**PRUEBAS UNITARIAS**	**27,5 da...**	**30/04/1...**	**6/06/13 14:00**	
84		LGAP_PRU_TPRE1	2,5 days	30/04/13 ...	2/05/13 14:00	76
85		LGAP_PRU_TPRE2	2,5 days	2/05/13 1...	7/05/13 9:00	84
86		LGAP_PRU_TPRE3	2,5 days	7/05/13 9...	9/05/13 14:00	85
87		LGAP_PRU_TPRE4	2,5 days	9/05/13 1...	14/05/13 9:00	86
88		LGAP_PRU_TPRE5	2,5 days	14/05/13 ...	16/05/13 14:00	87
89		LGAP_PRU_TPRE6	2,5 days	16/05/13 ...	21/05/13 9:00	88
90		LGAP_PRU_GNA	2,5 days	21/05/13 ...	23/05/13 14:00	89
91		LGAP_PRU_GDA	2,5 days	23/05/13 ...	28/05/13 9:00	90
92		LGAP_PRU_TPRE7	2,5 days	28/05/13 ...	30/05/13 14:00	91
93		LGAP_PRU_TPRE8	2,5 days	30/05/13 ...	4/06/13 9:00	92
94		LGAP_PRU_TPRE9	2,5 days	4/06/13 9...	6/06/13 14:00	93
95		**DOCUMENTACION. MEMORIA**	**60 days**	**6/06/13 ...**	**29/08/13 14:00**	**94**
96		LGAP_DOC_MEMORIA	45 days	6/06/13 1...	8/08/13 14:00	94
97		LGAP_DOC_MANUALES_USU	15 days	8/08/13 1...	29/08/13 14:00	96
98		**IMPLANTACION Y DESPLIEGUE**	**65 days**	**29/08/1...**	**28/11/13 14:00**	
99		LGAP_IMPL_PROG	35 days	29/08/13 ...	17/10/13 14:00	97
100		LGAP_IMPL_SIETTE	30 days	17/10/13 ...	28/11/13 14:00	99
101		PRESENTACION	15 days	28/11/13 ...	19/12/13 14:00	100
102		DEFENSA	1 day	19/12/13 ...	20/12/13 14:00	101
103		FIN	0 days	20/12/13 ...	20/12/13 14:00	102

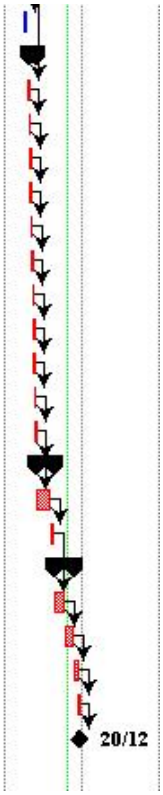


Figura 3.10: Diagrama de Gantt completo del proyecto (4/4).

para alcanzar los objetivos previstos en el Plan de Estudios, y se corresponde con una carga de trabajo del estudiante de 25 a 30 horas. Teniendo en cuenta que un proyecto de fin de carrera en la UNED tiene un carga lectiva de 6 créditos, el presente proyecto no debería llevar más de 180 horas de duración. Según podemos observar en la planificación realizada en el capítulo anterior, la duración del proyecto se aproxima a las 2.600 horas por lo que se supera con creces el mínimo exigido para un proyecto fin de carrera de estas características.

El alcance del presente proyecto vendrá dado por la cantidad de preguntas distintas que se quieran generar de forma automática. Una vez obtenido el conocimiento teórico sobre la materia a partir de la cuál se van a obtener las preguntas a generar, el siguiente paso es elegir un tipo concreto de pregunta y empezar a trabajar sobre ella.

Para cada pregunta o tipo de pregunta distinta que se quiera plantear, tendrán que realizarse una serie de acciones que quedan definidas por los requisitos funcionales identificados en las tablas 3.1 y 3.2. El número de tipos de preguntas distintas que se plantean en este proyecto es de 9. Cada tipo de pregunta permite con la parametrización adecuada, generar las instancias distintas que darán lugar a la generación posterior de cada una de las preguntas generadas por la librería. Por tanto la dimensión del proyecto viene dada

por el tamaño de la batería de preguntas ofrecido finalmente por la librería desarrollada. El requisito de escalabilidad definido en el proyecto para esta librería permitirá que se incorporen nuevos tipos de preguntas y hacer el proyecto tan largo como se quiera, incorporar en el futuro nuevos bloques temáticos o tipos de preguntas que permitan realizar ampliaciones al mismo.

El presente proyecto proporciona el *framework* básico para la definición e integración de preguntas de test que se generan de manera automática. A partir de ahí, para determinar el alcance real de este proyecto de fin de carrera se estima que una cantidad razonable de preguntas puede ser alrededor de treinta. Los bloques temáticos sobre los que van a tratar las preguntas generadas son los siguientes:

- relaciones de independencia probabilística y de independencia probabilística condicional que se plantean sobre una distribución de probabilidad.
- relaciones de independencia probabilística y de independencia probabilística condicional que se plantean sobre un grafo (dirigido o no dirigido).
- inferencia en redes bayesianas: típicas y de generación aleatoria.

La asignatura que se defina en el sistema *Siette* para demostrar el funcionamiento de la librería desarrollada, estará configurada por unas treinta preguntas distintas (ver apéndice A). A partir de la definición de las preguntas haciendo uso de la librería, se generarán los test de autoevaluación que se alimentan dichas preguntas para su confección.

Del mismo modo, el programa cliente que se desarrollará como un requisito más de este proyecto generará exámenes en formato PDF abasteciéndose de las preguntas generadas por la librería desarrollada.

Con este estudio queda perfectamente delimitado el alcance del proyecto y su viabilidad como proyecto de fin de carrera de una carrera de segundo ciclo como es la titulación de Ingeniero en Informática de la UNED.

Capítulo 4

Fundamentos

En este capítulo se va a realizar la exposición de los fundamentos o la base teórica sobre las que van a versar cada una de la preguntas que se van a generar de manera automática por la librería desarrollada en el presente proyecto. Cada uno de los apartados que se exponen a continuación, se corresponde con un bloque temático del dominio elegido para la generación automática de las preguntas. Todo el contenido teórico de este apartado se ha extraído y se puede ver completo de los apuntes (F.J.Díez, 2010)(F.J.Díez, 2013).

4.1. Relaciones de Independencia Probabilística

En este apartado nos centraremos en las relaciones de independencia probabilística que se pueden dar en un sistema en el que tenemos definidas una serie de variables aleatorias discretas y una distribución de probabilidad definida sobre todas las combinaciones posibles que se pueden dar entre los valores que toman cada una de esas variables aleatorias. En los siguientes aparatados se establecerán las definiciones formales de todos los conceptos que aparecerán en las preguntas planteadas y que son necesarios para la resolución de cada una de ellas.

4.1.1. Base teórica

En la generación de este tipo de preguntas nos vamos a centrar en las relaciones que se pueden establecer entre las variables aleatorias de un sistema. En primer lugar damos la definición formal de variable aleatoria como sigue:

Variable aleatoria.- aquella que toma valores que, a priori, no conocemos con certeza. Cada uno de estos valores que puede tomar la variable tendrá asociado un valor numérico, comprendido ente cero y uno, que representa la probabilidad de que la variable tome ese

valor.

El conjunto de valores que puede adoptar la variable es un conjunto discreto, se decir está formado por un conjunto finito de elementos y estos constituyen una distribución de probabilidad, es decir, la suma de los valores numéricos asociados a cada valor que puede tomar la variable aleatoria ha de ser la unidad. De ahora en adelante, cuando hablemos de variables aleatorias, entenderemos que nos referimos a variables aleatorias discretas.

Los sistemas que vamos a considerar para el planteamiento de nuestras preguntas, además de un conjunto de variables aleatorias discretas, se componen de una distribución de probabilidad definida sobre las distintas configuraciones que se pueden dar entre dichas variables. A partir de una configuración del sistema, es decir, de una distribución de probabilidad que asocia a cada combinación distinta de los valores de cada una de las variables del sistema un valor numérico, una probabilidad, nos remitimos a la definición de Probabilidad Conjunta que dice lo siguiente:

Probabilidad Conjunta .- Dados $X = \{X_1, ..., X_n\}$, conjunto de variables aleatorias discretas del sistema $x =(x_1, ..., x_n)$ configuraciones posibles, definimos probabilidad conjunta como la aplicación que a cada configuración le asigna un número real no negativo de modo que:

$$\sum_x P(X) = \sum_{x_1} ... \sum_{x_n} P(x_1, ..., x_n) = 1$$

Una vez establecidas las variables aleatorias de nuestro sistema y una distribución de probabilidad, consideraremos dos tipos de relaciones posibles entre las variables del sistema: la relación de i**ndependencia probabilística** y la **relación de independencia probabilística condicional.**

A continuación se dan las definiciones formales de ambos conceptos:

Definición de variables independientes probabilísticamente, denotado como $I_p(X, Y)$ Dadas dos variables aleatorias X e Y se dice que son independientes probabilísticamente si y sólo si

$$\forall x, \forall y P(x, y) = P(x) \times P(y)$$

Es decir para todas las combinaciones posibles de los valores que pueden tomar las variables X e Y se cumple que la probabilidad conjunta de los valores de cada configuración coincide con el producto de las probabilidad de que cada variable tome el valor correspondiente en la configuración dada, es decir la probabilidad marginal de cada variable que se define como:

Probabilidad Marginal .- dados $P(x_1, ..., x_n)$ $X' = \left\{X'_1, ..., X'_n\right\} \subset X$

$$P(x') = P(x'_1, ..., x'_n) = \sum_{x_i | x_i \notin x'} P(x_1, ..., x_n)$$

$$P(x_i) = \sum_{x_j | Xj \neq X_i} P(x_1, ..., x_n)$$

Como ejemplo supongamos que tenemos las variables aleatorias discretas $X = (x_0, x_1)$ e $Y = (y_0, y_1)$ con la distribución de probabilidad dada por la tabla 4.1:

Tabla 4.1: Ejemplo Distribución de probabilidad entre las variables X e Y.

X	Y	$P(X,Y)$
x_0	y_0	$0,15$
x_0	y_1	$0,35$
x_1	y_0	$0,30$
x_1	y_1	$0,20$

Para comprobar si entre las variables X e Y se da una relación de independencia probabilística, denotado como $I_p(X, Y)$, comprobaríamos, a partir de los datos de la tabla anterior, si se cumple que el valor de la probabilidad conjunta de cada fila $P(x_{i,}y_i)$ coincide con el producto de la probabilidad marginal de cada valor que toma cada variable. Es decir, tendríamos que comprobar si se cumple la siguiente expresión:

$$\forall x, \forall y P(x, y) = P(x) \times P(y)$$

Para comprobar si se cumple, construimos la siguiente tabla:

	y_0	y_1	$P(x)$
x_0	$0,15$	$0,35$	$0,5$
x_1	$0,30$	$0,20$	$0,5$
$P(y)$	$0,45$	$0,55$	1

A partir de los datos de la tabla del ejemplo vamos rellenando la tabla anterior realizando la suma de las distintas combinaciones en las que aparece cada valor binario concreto. Por ejemplo, para obtener el valor de la probabilidad del valor concreto x_0 (probabilidad marginal), sumamos los valores correspondientes de las dos primeras filas de la tabla del

ejemplo, prescindiendo del valor genérico y (es decir y_0 e y_1) y obtenemos el resultado $0,15+0,35=0,5$. Del mismo modo vamos rellenando el resto de filas y columnas. Observar que la última fila y la última columna corresponde precisamente a las probabilidades marginales.

Observando la tabla del ejemplo comprobamos, que la configuración (x_0, y_0) no cumple con la condición de independencia pues no se verifica la siguiente igualdad:

$$P(x_0, y_0) \neq P(x_0) \times P(y_0)$$

como podemos comprobar sustituyendo por los correspondientes valores numéricos:

$$0,15 \neq 0,5 \times 0,45$$

$$\boxed{\mathbf{0,15 \neq 0,225}}$$

Por tanto X y Y no son independientes probabilísticamente, no se cumple la relación $I_p(X,Y)$ y concluimos que no se da una relación de independencia probabilística entre las variables o que las variables X e Y no son independientes probabilísticamente, pues hemos encontrado una combinación para la cuál no se cumple la definición dada en 4.1.1

Hemos comprobado que no se cumple la independencia probabilística entre X y Y para el caso x_0y_0, pero en la siguiente tabla podemos comprobar que no se cumple para ningún caso, es decir $\forall x, \forall y\, P(x,y) \neq P(x) \times P(y)$:

		$P(x,y)$	$\neq$	$P(x)$	$P(y)$	$P(x,y) \neq P(x).P(y)$:
x_0	y_0	$0,15$	$\neq$	$0,5$	$0,45$	$0,15 \neq 0,225$
x_0	y_1	$0,35$	$\neq$	$0,5$	$0,55$	$0,35 \neq 0,275$
x_1	y_0	$0,3$	$\neq$	$0,5$	$0,45$	$0,3 \neq 0,225$
x_1	y_1	$0,2$	$\neq$	$0,5$	$0,55$	$0,2 \neq 0,275$

A continuación vamos a dar la definición del otro tipo de relación entre variables aleatorias que vamos a considerar, se trata de la relación de independencia probabilística condicional.

Tabla 4.2: Ejemplo distribución de probabilidad sobre las variables binarias A B y C.

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	$0,03$
$+a$	$+b$	$\neg c$	$0,12$
$+a$	$\neg b$	$+c$	$0,07$
$+a$	$\neg b$	$\neg c$	$0,28$
$\neg a$	$+b$	$+c$	$0,03$
$\neg a$	$+b$	$\neg c$	$0,27$
$\neg a$	$\neg b$	$+c$	$0,02$
$\neg a$	$\neg b$	$\neg c$	$0,18$

Definición de variables condicionalmente independientes probabilísticamente denotado por $I_p(X,Y \mid Z)$:

$$\forall x, \forall y, \forall z\, P(z) > 0 \Rightarrow P(x,y \mid z) = P(x \mid z) \times P(y \mid z)$$

Es decir tendríamos que comprobar que para todas las configuraciones posibles de los valores de las variables se cumple la expresión anterior. Para desarrollar la expresión anterior damos a continuación la definición de probabilidad condicional:

Probabilidad condicional.- dados $X = \{X_1, ..., X_n\}$ e $Y = \{Y_1, ..., Y_m\}$ conjuntos disjuntos y una configuración x de X tal que $P(x) > 0$

$$P(y \mid x) = \frac{P(x,y)}{P(x)}$$

en general $P(y \mid x) \neq P(x \mid y)$

Proposición.- Dados dos subconjuntos disjuntos de variables X e Y y una configuración x de X tal que $P(x) > 0$ se cumple que:

$$\forall_x \sum_y P(y \mid x) = 1$$

En el ejemplo siguiente se da una distribución de probabilidad P dada por la tabla 4.1:

En ella hemos considerado un tipo particular de variable aleatoria, la variable aleatoria binaria que se define como aquella que únicamente toma dos posibles valores denotados

como $X = (+x, \neg x)$. Para comprobar si se cumple por ejemplo la relación $I_p(A, B \mid C)$ entre las variables A, B y C dadas en el ejemplo, comprobaríamos si se verifica la siguiente expresión:

$$\forall a, \forall b, \forall c\, P(c) > 0 \Rightarrow P(a, b \mid c) = P(a \mid c) \times P(b \mid c)$$

Comprobamos que, por ejemplo, para la configuración$(+a, +b, +c)$ no se cumple la relación pues no se cumple la igualdad:

$$P(+a, +b \mid +c) \neq P(+a \mid +c) \times P(+b \mid +c)$$

$$\frac{P(+a, +b, +c)}{P(+c)} \neq \frac{P(+c, +a)}{P(+c)} \times \frac{P(+c, +b)}{P(+c)}$$

$$\frac{0,03}{0,5} \neq \frac{0,1}{0,15} \times \frac{0,06}{0,15}$$

$$0,2 \neq 0,6666667 \times 0,4$$

$$0,2 \neq 0,2666667$$

Se concluye que las variables no son condicionalmente independientes probabilísticamente, y por tanto no se cumple la relación $I_p(A, B \mid C)$ es decir que no se da una relación de independencia probabilística condicional entre las variables A y B respecto de C o que las variables A y B no son independientes condicionalmente dado C, pues hemos encontrado una configuración que no verifica la definición dada 4.1.1.

Una vez dadas las definiciones anteriores, tenemos la base teórica necesaria para dar respuesta a las preguntas que se pueden plantear y que tienen que ver con la relaciones tanto de independencia probabilista entre dos variables como de independencia probabilista condicional de un conjunto de variables respecto de otra (o de un par de variables respecto de una tercera). Esta es la materia para las preguntas del bloque temático 1 del dominio.

4.2. Relaciones de Independencia Probabilística sobre Grafos

En este apartado se plantearán diversas preguntas relacionadas con la teoría de grafos conjuntamente con las relaciones de independencia probabilística entre variables aleatorias discretas de un sistema como paso previo a la definición de redes Bayesianas que se plantean en el siguiente bloque temático.

4.2.1. Base teórica

En este apartado se establecerán las nociones básicas sobre la teoría de grafos. Consideramos las siguientes definiciones:

- **Grafos.**- Conjunto de nodos (N) y enlaces.
- **Enlaces**.- dado un conjunto de nodos $\mathcal{N}$, un enlace es una terna de $\mathcal{N}x\mathcal{N}x\{dirigido, no-dirigido\}$ es decir $\{X, Y, d\}$

$$\{X, Y, dirigido\} \Longleftrightarrow X \longrightarrow Y$$

$$\{X, Y, no-dirigido\} \Longleftrightarrow X - Y$$

- **Grafo**.- Es un par $\mathcal{G} = (\mathcal{N},\mathcal{A})$ con $\mathcal{N}$conjunto de nodos y $\mathcal{A}$ conjunto de enlaces.
- **Grafo dirigido**.- los enlaces son todos dirigidos.
- **Grafo no dirigido**.- los enlaces son todos NO dirigidos.
- **Grafo mixto o híbrido**.- los enlaces son dirigidos o no dirigidos.
- **Padre**.- X es padre de Y sii $\exists$ arco $X \longrightarrow Y$ se denota $Pa(X)$.
- **Hijo**.- Y es hijo de X sii $\exists$ arco $X \longrightarrow Y$.
- **Hermano**.- X es hermano de Y sii $\exists$arco $X - Y$
- **Antepasado**.- X es antepasado de Z sii $\exists$ al menos un nodo Y tal que X es padre de Y e Y antepasado de Z.
- **Descendiente**.- Z es descendiente de X sii X es antepasado de Z.

- **Camino**.- Un camino entre X_1 y X_n es una sucesión de nodos diferentes $\{X_1, ..., X_n\}$ pertenecientes a una grafo $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$ tal que para cada par de nodos consecutivos $\exists$ un enlace:

$$\forall i, 1 \leq i \leq N, (X_i \rightarrow X_{i+1}) \in A \vee (X_{i+1} \rightarrow X_i) \in A \vee (X_i - X_{i+1}) \in A$$

- **Camino dirigido**.- Un camino dirigido desde X_1 a X_n es una sucesión de nodos diferentes $\{X_1, ..., X_n\}$ pertenecientes a una grafo $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$ tal que para cada par de nodos consecutivos $\exists$ un enlace dirigido del primero al segundo:

$$\forall i, 1 \leq i \leq N, (X_i \rightarrow X_{i+1}) \in A$$

- **Grafo conexo**.- si entre dos cualesquiera de sus nodos hay al menos un camino.
- **Grafo simplemente conexo**.- si entre cada par de nodos existe un único camino.
- **Grafo multiplemente conexo**.- si contiene al menos un par de nodos entre los cuales hay más de un camino.
- **Ciclo en grafo dirigido**.- sucesión de nodos diferentes $\{X_1, ..., X_n\}$ pertenecientes a una grafo $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$ tal que entre cada par de nodos consecutivos $\exists$ un enlace dirigido del primero al segundo, $X_i \rightarrow X_{i+1}$ y $\exists$ además un encale $X_n \rightarrow X_1$.
- **Bucle**.- sucesión de nodos diferentes $\{X_1, ..., X_n\}$ pertenecientes a una grafo $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$ tal que entre cada par de nodos consecutivos $\exists$ un enlace y existe además un enlace entre el primero y el último, y no es un ciclo.
- **Ciclo en grafo no dirigido**.- sucesión de nodos diferentes $\{X_1, ..., X_n\}$ pertenecientes a una grafo no dirigido $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$ tal que:

  1. $Xi \neq Xj$ para $1 \leq i < j \leq N$
  2. $\forall i < N, \exists A$ un arco (Xi, X_{i+1})
  3. $\exists$ además un arco $(X_{n,}X_1)$

4.3. Inferencia en Redes Bayesianas

En este apartado se plantearán diversas preguntas relacionadas con las redes Bayesianas, sobre algunas conocidas y que suelen ser objeto de estudio y sobre otras de generación aleatoria. Esta tercera parte será la materia sobre la que se plantearán las preguntas del tercer bloque temático del dominio.

4.3.1. Base teórica

Partiremos en este apartado con la presentación del teorema de Bayes (McGrayne, 2012) fundamento sobre el que se sustenta la definición de una red bayesiana.

- **Teorema de Bayes**.- Dadas dos configuraciones x e y de dos subconjuntos de variables X e Y, respectivamente, tales que $P(x) > 0$ y $P(y) > 0$ se cumple que:

$$P(x \mid y) = \frac{P(x) \times P(y \mid x)}{\sum_{x' \mid P(x') > 0} P(x') \times P(y \mid x')}$$

- **Proposición**.- Dados tres subconjuntos disjuntos X, Y,Z si $P(y, z) > 0$ se cumple que:

$$P(x, y \mid z) = P(x \mid y, z) \times P(y \mid z)$$

- **Teorema de Bayes con condicionamiento**.- Dadas tres configuraciones x , y, z de tres subconjuntos de variables X e Y, Z respectivamente, tales que $P(x, z) > 0$ y $P(y, z) > 0$ se cumple que:

$$P(x \mid y) = \frac{P(x \mid z) \times P(y \mid x, z)}{\sum_{x' \mid P(x' \mid z) > 0} P(y \mid x', z) \times P(x' \mid z)}$$

Aplicación del teorema de Bayes

- **Hallazgo**.- Determinación del valor de una variable H=h, a partir de un dato (observación, medida, etc).

- **Evidencia**.- Conjunto de todos los hallazgos disponibles en un determinado momento o situación $e = \{H_1 = h_1, ..., H_r = h_r\}$.

- **Probabilidad a priori**.- probabilidad de una variable o subconjunto de variables cuando no hay ningún hallazgo.

  "La probabilidad a priori de X coincide con la probabilidad marginal $P(x)$"

- **Probabilidad a posteriori**.- probabilidad de una variable o subconjunto de variables dada la evidencia e: $P(x \mid e)$.

  Teorema de Bayes → Problemas de Clasificación → Diagnóstico (caso particular)

- **Forma normalizada del Teorema de Bayes**.-

$$P(x \mid y) = \alpha P(x) \times P(y \mid x)$$

$$\alpha = \left[\sum_{x'} P\left(x'\right) \times P\left(y \mid x'\right)\right]^{-1} = \left[P(y)\right]^{-1}$$

ponderación de probabilidad a priori ↔ verosimilitud

- **Forma racional del Teorema de Bayes**.-

$$\frac{P(x^i \mid y)}{P(x^j \mid y)} = \frac{\alpha \times P(x^i) \times P(y \mid x^i)}{\alpha \times P(x^j) \times P(y \mid x^j)} = \frac{P(x^i)}{P(x^j)} \times \frac{P(y \mid x^i)}{P(y \mid x^j)}$$

$\frac{P\left(x^i\right)}{P(x^i)}$ razón de probabilidad

$\frac{P\left(y|x^i\right)}{P(y|x^j)}$ razón de verosimilitud

Variables binarias

- Razón de probabilidad de X a priori:

$$RP_{pre}(X) \equiv \frac{P(+x)}{P(\neg x)} = \frac{P(+x)}{1 - P(+x)}$$

- Razón de probabilidad de X a posteriori:

$$RP_{post}(X) \equiv \frac{P(+x \mid y)}{P(\neg x \mid y)} = \frac{P(+x \mid y)}{1 - P(+x \mid y)}$$

- Razón de verosimilitud para X dado y

$$RVx(y) \equiv \frac{P(y \mid +x)}{P(y \mid \neg x)}$$

- Expresiones

$$P(+x \mid y) = \frac{RP_{post}(x)}{1 + RP_{post}(x)}$$

$$RP_{post}(x) = RP_{pre}(x).RVx(y)$$

Prevalencia	$P(+x)$
Sensibilidad	$P(+y \mid +x)$
Especificidad	$P(\neg y \mid \neg x)$
Valor Predictivo Positivo (VPP)	$P(+x \mid +y)$
Valor Predictivo Negativo (VPN)	$P(\neg x \mid \neg y)$

- VPP aumenta considerablemente al aumentar la especificidad→ para confirmar enfermedad buscar pruebas especificas.
- VPN aumenta al aumentar la sensibilidad → para descartar enfermedad, buscar síntomas o signos muy sensibles.

Método Bayesiano Ingenuo Suposiciones (hipótesis):

1. diagnósticos exclusivos y exhaustivos.
2. independencia condicional: los hallazgos son condicionalmente independientes entre si para cada diagnóstico.

$$P(d \mid h_1, ..., h_m) = \alpha P(d) \times P(h_1 \mid d) \times ... \times P(h_m \mid d)$$

Forma Racional del Método Bayesiano Ingenuo

$$\frac{P(d \mid h_1, ..., h_m)}{P(d' \mid h_1, ..., h_m)} = \frac{P(d)}{P(d')} \times \frac{P(h_1 \mid d)}{P(h_1 \mid d')} \times ... \times \frac{P(h_m \mid d)}{P(h_m \mid d')}$$

Las Redes Bayesianas y los Diagramas de influencia son Grafos Dirigidos Acíclicos (GDA).

Grafos Dirigidos Acíclicos (GDA).

- **Familia** .- conjunto formado por X y los padres de X:

$$Fam(X) = \{X\} \cup Pa(X)$$

- **Nodo terminal**.- nodo que no tiene hijos.
- **Poliárbol**.- grafo dirigido simplemente conexo, es decir, un grafo dirigido que no contiene ciclos ni bucles.
- **Árbol**.- es un caso particular de poliárbol, en que cada nodo tiene un sólo padre, excepto el nodo raíz, que no tiene padres.

- **Ordenación ancestral de un GDA**.- todos los antepasados de cada nodo son mayores que él.
- **Proposición**.- En todo GDA $\exists$ al menos un nodo sin padres.
- **Proposición**.- Todo GDA tiene al menos un orden ancestral.
- **Proposición**.- Cuando los nodos de un grafo están numerados, por ejemplo $\{X_1, ..., X_n\}$ $\exists$ una permutación α tal que $X_{\alpha(1)} > ... > X_{\alpha(n)}$ es una ordenación ancestral.

Definición de Red Bayesiana Consta de tres elementos:

1. Conjunto de variables aleatorias X.
2. GDA $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$ grafo dirigido acíclico en el que cada nodo representa una variable X_i.
3. distribución de probabilidad X, $P(X)$ factorizada así:

$$P(X) = \prod_i P(x_i \mid pa(X_i))$$

donde las $P(x_i \mid pa(X_i))$ son las probabilidades condicionadas que se obtienen a partir de $P(X)$.

El método probabilísta clásico es un caso particular de red bayesiana:

$$P(d \mid h_1, ..., h_m) = P(d) \times P(h_1 \mid d) \times ... \times P(h_m \mid d)$$

Construcción de una Red Bayesiana Sea un conjunto de variables X, un GDA $\mathcal{G}$ en que cada nodo representa una variable de X y un conjunto de distribuciones de probabilidad $\{P_c(x_i \mid pa(X_i))\}$, es decir, una distribución por cada variable X_i y por cada configuración de sus padres en el grafo. La función $P(X)$, definida por

$$P(x) = \prod_i P_c(x_i \mid pa(X_i))$$

es una distribución de probabilidad. Se cumple además que:

$$\forall i, \forall x_i, \forall pa(X_i), P(x_i \mid pa(X_i)) = P_c(x_i \mid pa(X_i))$$

Propiedad de Markov Una terna (X, G, P) cumple la propiedad sii

$$P\left(x \mid pa\left(x\right), y\right) = P\left(x \mid pa\left(x\right)\right)$$

- **Teorema**.- Toda terna (X, G, P) que cumple la propiedad de Markov constituye una Red Bayesiana.
- **Teorema**.- Toda terna (X, G, P) que constituye una Red Bayesiana cumple la propiedad de Markov.

Grafos de dependencias e independencias probabilistas

- **Camino activo en un grafo no dirigido**.- Sea un grafo no dirigido $\mathcal{G}$ dos nodos A y B y un subconjunto C tal que ni A ni $B \in C$.
  - un camino de dos nodos $A - B$ (un sólo enlace) siempre está activo.
  - un camino de n nodos $A - X_1 - ... - X_{n-2} - B$ está **activo** cuando ningún nodo entre A y B pertenece a C, es decir, $\{X_1, ..., X_n\} \cap C = \emptyset$.
- **Camino activo en un grafo dirigido**.- Sea un grafo dirigido $\mathcal{G}$ dos nodos A y B y un subconjunto C tal que ni A ni $B \in C$.
  - un camino de dos nodos $A \longrightarrow B$ o $B \longrightarrow A$ (un sólo enlace) siempre está activo.
  - un camino de tres nodos puede ser: $A \longrightarrow X \longrightarrow B$, $B \longrightarrow X \longrightarrow A$, $A \longleftarrow X \longrightarrow B$o $A \longrightarrow X \longleftarrow B$.
  - un camino del tipo $A \longrightarrow X \longrightarrow B$ o $A \longleftarrow X \longrightarrow B$ está activo si $X \notin C$
  - un camino del tipo $A \longrightarrow X \longleftarrow B$ está activo si X o alguno de sus descendientes $\in C$.
  - un camino de n nodos está activo si cada par de enlaces consecutivos forman un camino activo.
- **Nodos conectados**.- Sea un grafo dirigido o no $\mathcal{G}$ y un subconjunto C de $\mathcal{G}$. Dos nodos de $\mathcal{G}$ están conectados (dado C) cuando $\exists$ al menos un camino activo entre ellos. En otro caso se dice que están separados.

$$I_{\mathcal{G}}\left(A, B \mid C\right) \text{ indica } A \text{ y } B \text{ separados.}$$
$$\neg I_{\mathcal{G}}\left(A, B \mid C\right) \text{ indica } A \text{ y } B \text{ conectados.}$$

- **Subconjuntos Conectados**.- Dos subconjuntos de nodos A y B están conectados dado C cuando $\exists$ al menos un nodo de A conectado con un nodo de B dado C y se expresa mediante $\neg I_{\mathcal{G}}(A, B \mid C)$ en otro caso se dice que están separados $I_{\mathcal{G}}(A, B \mid C)$.

Mapas de independencias Dado una terna (X, G, P) es *I-map* de P si para todo trío de subconjuntos $\{A, B, C\}$ de X se cumple:

$$I_{\mathcal{G}}(A, B \mid C) \Rightarrow P(a, b \mid c) = P(a \mid c) \times P(b \mid c)$$

$$I_{\mathcal{G}}(A, B \mid C). \Rightarrow I_p(A, B \mid C)$$

- **Teorema**.- El grafo de una Red Bayesiana constituye un Mapa de Independencias de la distribución de probabilidad de dicha red.
- **Teorema** .- Una terna (X, G, P) tal que $\mathcal{G}$ es un Mapa de Independencias de$P$, constituye una Red Bayesiana.

Capítulo 5

Análisis

En este capítulo vamos a comentar la etapa o fase de análisis del proyecto, partiendo del desglose de tareas de análisis realizado en el apartado de planificación del proyecto (ver 3.2.1). En una primera instancia realizaremos un estudio previo de cada componente que va a tener nuestro sistema y de la división realizada en los tres bloques temáticos establecidos. En el apartado siguiente identificamos cada uno de los componentes de nuestro sistema.

5.1. Componentes del sistema

5.1.1. Características comunes

La primera tarea identificada en el plan del proyecto es ***LGAP_ANA_COMUN***. En este punto vamos a identificar las características comunes que van a tener las distintas preguntas o tipos de preguntas planteadas. A partir de ahora hablaremos de tipos de preguntas como un tipo concreto de pregunta de la que se crearan distintas instancias cada una de ellas resultado de una determinada parametrización. Se trata de preguntas cortas de test que están compuestas por un enunciado que plantea la pregunta y una serie de respuestas o alternativas a la pregunta planteada en el enunciado. El usuario de la librería podrá elegir el número de respuestas o alternativas que se plantean para cada enunciado y cuántas de las mismas serán respuestas correctas. Identificamos pues, en este punto, los dos primeros parámetros que nos permitirán definir distintas instancias de un mismo tipo de pregunta como son: el número de respuestas o alternativas a la pregunta planteada y cuántas de ellas serán correctas. Por tanto tendremos en común para todas las preguntas los siguientes elementos:

- enunciado de la pregunta, el enunciado consistirá en un texto fijo (cadena de caracteres) en el que además existirá un parte variable que se genera automáticamente y que es la que permitirá tener distintas preguntas (o instancias de la pregunta) del mismo tipo.

- alternativas o posibles respuestas a la pregunta que se plantea en el enunciado. De todas estas alternativas, en función de la parametrización, el alumno deberá elegir cuál o cuales de las respuestas posibles es correcta.

Por tanto y a groso modo podemos tener ya un primer componente común que desarrollaremos posteriormente en nuestro sistema y cuyo diagrama de componentes corresponde al de la figura 5.1.

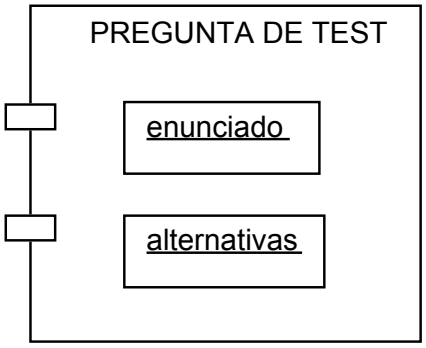


Figura 5.1: Componente que define una pregunta de test.

El componente que implemente cada tipo o familia de pregunta deberá proporcionar la funcionalidad de marcar la respuesta o respuestas correctas a la pregunta planteada en el enunciado y proporcionar la información de refuerzo o explicación de como se ha llegado a la conclusión de que una respuesta es correcta o incorrecta.

El siguiente paso será establecer un plan para cada tipo distinto de pregunta que se quiera plantear. En este proyecto, la librería generada planteará nueve tipos distintos de familias o tipos de pregunta. Cada uno de estos tipos de pregunta pertenecerá a uno de los tres bloques temáticos definidos en el dominio. Teniendo esto en cuenta, los tipos de pregunta de 1 a 4 tratarán del bloque temático presentado en el punto 4.1, preguntas de tipo 5 y 6 corresponderán al bloque temático especificado en el punto 4.2 y por último los tipos de preguntas de 7 a 9 y se plantean en base al bloque temático 4.3.

En los siguientes apartados, se explicará en detalle en qué consisten cada uno de estos tipos de pregunta.

5.1.2. Relaciones de Independencia Probabilística

En este apartado se va a analizar como se puede abordar la realización de un programa informático que sea capaz de generar preguntas basadas en la base teórica presentada en el punto 4.1.1 del capítulo anterior. Para ello vamos a dividir el problema en dos partes, por un lado la generación de enunciados, identificando la parte fija de la parte variable en la generación, aquella que se va a generar de forma automática. Por otro lado, para la resolución del ejercicio, determinaremos qué acciones hay que realizar para dar con las soluciones a los apartados que se plantean. El programa debe presentar además de la solución correcta a la pregunta planteada, los pasos que se han seguido para la resolución de la pregunta, es decir el refuerzo.

Todos los tipos de preguntas se plantean a partir de un sistema compuesto por un conjunto de variables aleatorias y una distribución de probabilidad. Los distintos tipos de preguntas se plantearán en base a unas relaciones de independencia probabilística o de independencia probabilística condicional que se establecen entre las variables del sistema a partir de la distribución de probabilidad dada. Estas relaciones se establecerán de manera aleatoria.

5.1.2.1. Generación del enunciado

Para la generación del enunciado se seguirán los siguientes pasos:

1. **Elegir el número de variables aleatorias que formaran parte del sistema así como los distintos valores que pueden tomar cada una de ellas.** Este número no puede ser muy grande por el contexto en el que estamos trabajando, una pregunta de test de un examen o ejercicio de evaluación. Un número razonable, podría ser un valor comprendido entre 2 y 5 ya que un valor superior a 5 haría el ejercicio largo y tedioso y un valor inferior a 2 nos impide plantear el problema, ya que necesitamos hablar de probabilidad conjunta de al menos dos variables para poder establecer relaciones de independencia probabilística. En cuanto al tipo de variables aleatorias que se van a utilizar y los valores que pueden tomar, hay que definirlos de forma exclusiva y exhaustiva de acuerdo a la definición de variable aleatoria. Si utilizamos variables aleatorias binarias, tendremos dos valores posibles representados genéricamente por $+x$ y $\neg x$. En el caso de la tabla 4.2 tenemos tres variables aleatorias binarias representadas por A, B, C y los valores posibles que pueden tomar vienen representados por los valores genéricos a, b, c respectivamente. Los valores concretos de a vendrán dador por $+a$ y $\neg a$, los de b por $+b$ y $\neg b$ y los

de c por $+c$ y $\neg c$. Si tenemos en cuenta que cada variable aleatoria representa un conjunto, aquel formado por los distintos valores que puede tomar la variable aleatoria, el número de configuraciones posibles o configuraciones será el producto cartesiano de todos conjuntos representados por cada una de las variables aleatorias. En general, si tenemos un conjunto de variables aleatorias $X = \{X_1, ..., X_n\}$ cada una de las configuraciones $x =(x_1, ..., x_n)$ será un elemento del producto cartesiano de las variables. Su cardinalidad vendrá dada por el producto $\mid X_1 \mid \cdot ... \cdot \mid X_n \mid$, es decir por el producto de las cardinalidades de cada conjunto representado por cada variable aleatoria. En el caso particular de las variables aleatorias binarias, al poder tomar únicamente dos valores distintos, la cardinalidad de cada conjunto que representa cada variable es 2 y por tanto la cardinalidad del producto cartesiano viene dada por la expresión 2^n siendo n el número de variables aleatorias que estamos considerando.

2. **Elegir el tipo de las variables que forman parte del sistema**. Tendremos tres posibilidades:

   a) todas las variables aleatorias del sistema son binarias, es decir únicamente pueden tomar dos valores posibles: $(\neg x, +x)$.

   b) todas las variables aleatorias del sistema son genéricas, es decir pueden tomar cualquier número de valores.

   c) el sistema puede estar compuesto tanto de variables aleatorias binarias como genéricas.

3. En este punto nos centraremos en implementar un **generador de distribuciones de probabilidad** (que llamaremos G). El problema a resolver consistirá en determinar los parámetros que son necesarios generar para llegar a obtener el conjunto formado por los N números que constituyen la distribución de probabilidad. Un parámetro importante más a tener en cuenta es la precisión a utilizar en los cálculos. La precisión, es decir el número de decimales que vamos a considerar puede influir en que se cumpla o no una determinada relación. Nuestro generador será una función que a cada configuración le asigna un número real no negativo, comprendido entre 0 y 1 $(0 < x < 1)$ de modo que:

$$\sum_x P(X) = \sum_{x_1} ... \sum_{x_n} P(x_1, ..., x_n) = 1$$

4. Realizar un programa que **genere las distintas combinaciones** entre los valores de las variables fruto del producto cartesiano de las mismas y les asocie a cada combinación (configuración) el valor correspondiente generado por el generador de distribuciones de probabilidad G implementado en el punto anterior. La tabla tendrá tantas filas como combinaciones distintas, configuraciones, tengamos de las variables consideradas. La distribución de probabilidad generada no se elige al azar, los valores obtenidos se obtienen exigiendo que se cumpla una determinada propiedad de independencia probabilística; para conseguirlo elegimos una de las variables del sistema a partir de la cuál forzaremos la generación en el generador G para que se cumpla la relación de independencia probabilística respecto de la variable elegida. En este paso deberemos diseñar un algoritmo que nos permite realizar la generación cumpliendo esta característica de independencia probabilística.

5. Una vez generada la distribución de probabilidad, el siguiente paso será plantear las distintas relaciones entre las variables del sistema que se van a plantear en la pregunta. Empezaremos definiendo relaciones entre dos variables y posteriormente entre dos variables y condicionada a una tercera para las relaciones de independencia probabilística condicional. Para ello se pasa a **generar las combinaciones entre las distintas variables consideradas** para plantear las relaciones, en este caso de independencia probabilista entre dos variables $I_p(X, Y)$. Generaremos tantas combinaciones distintas como apartados de este tipo se quieran plantear hasta un máximo que vendrá dado por el número de variables que estemos considerando. Un ejemplo de pregunta podría contener las siguientes relaciones: $I_p(A, B)$, $I_p(A, C)$, $I_p(B, C)$.

6. Por último, se generarán las distintas combinaciones de 2 variables respecto de una tercera variable para plantear las relaciones de independencia probabilista condicional $I_p(X, Y \mid Z)$. Un ejemplo de combinaciones generadas podrían ser las siguientes: $I_p(A, B \mid C)$, $I_p(B, C \mid A)$.

Llegados a este punto ya tenemos todos los elementos necesarios para la generación de los distintos tipos de pregunta que tengan que ver con el planteamiento de relaciones de independencia probabilística o de independencia probabilística condicional entre las variables aleatorias de un sistema a partir de una distribución de probabilidad.

En el apartado anterior identificábamos 2 parámetros: el número de respuestas o alternativas que se plantean así como cuántas eran respuestas correctas. Con lo que se ha expuesto en los puntos anteriores, y a modo de recopilación, en total tendremos los

siguientes parámetros en un pregunta de este tipo:

- número de respuestas o alternativas que se plantean.
- número de respuestas correctas.
- precisión (número de decimales a utilizar).
- número de variables que componen el sistema.
- tipo de las variables del sistema.
- variable sobre la que se fuerza el generador G para que se verifique una o determinada relación de independencia probabilística.

5.1.2.2. Generación de la Solución

Una vez generado el enunciado de la pregunta, el programa ha de ser capaz de dar con la solución del mismo. Seguidamente explicaremos las acciones a realizar por el algoritmo que calcula la solución a cada alternativa planteada en el enunciado. En general comprobaremos si se verifican o no las relaciones de independencia entre las variables aleatorias consideradas.

- Para comprobar si una relación de independencia probabilística entre dos variables $I_p(X, Y)$ es verdadera o falsa tendremos que recurrir a la definición de variables independientes probabilísticamente**:** X e Y son independientes sii $\forall x, \forall y P(x, y) = P(x) \times P(y)$ ver 4.1.1.

Para comprobar si se cumple, construimos la siguiente tabla:

	$+y$	$\neg y$	$P(x)$
$+x$	$P(+x, +y)$	$P(+x, \neg y)$	$P(+x)$
$\neg x$	$P(\neg x, +y)$	$P(\neg x, \neg y)$	$P(\neg x)$
$P(y)$	$P(+y)$	$P(\neg y)$	1

A partir de los datos de la tabla del enunciado que hayamos generado, vamos rellenando los huecos de la tabla anterior realizando la suma de las distintas combinaciones en las que aparece cada valor binario concreto.

En el momento que encontremos una configuración (x, y) que no cumple con la condición de independencia:

$$P(x,y) \neq P(x) \times P(y)$$

Concluiremos que X e Y no son independientes probabilísticamente, no se cumple por tanto la relación $I_p(X,Y)$ y por tanto la relación es FALSA.

Si llegamos al final del proceso y no hemos encontrado ninguna configuración que no satisfaga la propiedad concluiremos que la relación es VERDADERA.

- Para los apartados de relación de independencia condicional $I_p(X,Y \mid Z)$ ver 4.1.1 nos lleva a la definición de variables condicionalmente independientes probabilísticamente: $\forall x, \forall y, \forall z\, P(z) > 0 \Rightarrow P(x,y \mid z) = P(x \mid z) \times P(y \mid z)$. Entraremos en un proceso en el que en cada paso elegiremos una configuración de todas las posibles entre los valores de las tres variables que formen parte de la expresión.

Si encontramos alguna combinación (configuración) que no satisfaga la igualdad anterior concluiremos que la relación es FALSA.

Si llegamos al final del proceso y se cumple la igualdad para todas las configuraciones posibles, es decir si se cumple $\forall x, \forall y, \forall z$ se concluye que las variables X,Y son condicionalmente dependientes respecto de Z.

Por tanto la relación $I_p(X,Y \mid Z)$ se cumple y es VERDADERA.

Veamos a continuación cuál sería el procedimiento a seguir con detalle:

1. Generar todas las posibles combinaciones o configuraciones de las variables que intervienen en la expresión:

x	y	z	$P(x,y,z)$
$+x$	$+y$	$+z$	$P(+x,+y,+z)$
$+x$	$+y$	$\neg z$	$P(+x,+y,\neg z)$
$+x$	$\neg y$	$+z$	$P(+x,\neg y,+z)$
$+x$	$\neg y$	$\neg z$	$P(+x,\neg y,\neg z)$
$\neg x$	$+y$	$+z$	$P(\neg x,+y,\neg z)$
$\neg x$	$+y$	$\neg z$	$P(\neg x,+y,\neg z)$
$\neg x$	$\neg y$	$+z$	$P(\neg x,\neg y,+z)$
$\neg x$	$\neg y$	$\neg z$	$P(\neg x,\neg y,\neg z)$

2. Para cada configuración, fila de la tabla anterior, comprobar si se cumple la expresión $P(x,y \mid z) = P(x \mid z) \times P(y \mid z)$ sustituyendo los valores genéricos por los concretos. Por ejemplo para la fila primera se comprobaría la expresión:

$$P(+x, +y \mid +z) = P(+x \mid +z) \times P(+y \mid +z)$$

Que daría lugar a:

$$\frac{P(+x, +y, +z)}{P(+z)} = \frac{P(+z, +x)}{P(+z)} \times \frac{P(+z, +y)}{P(+z)}$$

Todos los términos de la expresión anterior se pueden resolver a partir de los datos de la tabla generada en el punto 1 y comprobar si se cumple o no la igualdad.

Si encontramos una configuración que no cumple la igualdad, ya podemos concluir que la relación es FALSA.

Si llegamos a la última combinación de la tabla y la igualdad se ha cumplido para todos los casos, la expresión es VERDADERA.

Llegados a este punto ya tenemos analizado el problema y hemos establecido un plan para poder realizar posteriormente el diseño de una solución que nos permita conseguir los objetivos marcados. Además tenemos el conocimiento necesario para implementar la lógica que nos permite obtener la solución a cada pregunta planteada. En la figura 5.2 representamos el diagrama de componentes correspondiente al componente software que implemente este bloque temático. El desarrollo consistirá en definir un sistema formado por un conjunto de variables aleatorias y una distribución de probabilidad definida sobre dichas variables. A partir del sistema definido se plantean las distintas relaciones de independencia probabilística e independencia probabilística condicional entre las variables del sistema.

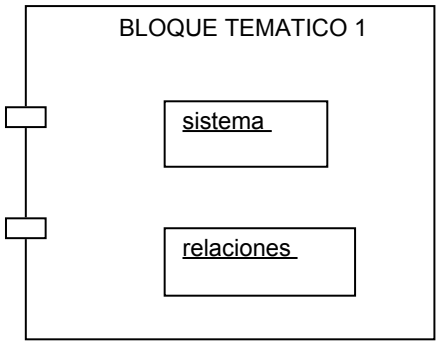


Figura 5.2: Componente que define una pregunta del bloque temático 1.

5.1.3. Grafos

En este apartado, al igual que hicimos en el apartado anterior, vamos a realizar el mismo análisis. Vamos a considerar relaciones de independencia probabilística o independencia probabilística condicional entre las variables aleatorias de un sistema. En este caso vamos a considerar grafos en lugar de distribuciones de probabilidad para plantear las relaciones. Trabajaremos con dos tipos de grafos acíclicos: grafos no dirigidos (GNA) y grafos dirigidos (GDA).

De nuevo vamos a dividir el problema en dos partes, por un lado la generación del enunciado, identificando la parte fija de la parte variable, aquella que se va a generar de forma automática. Por otro lado, para la resolución del ejercicio, determinaremos qué acciones hay que realizar para dar con las soluciones a los apartados que se plantean, que en el caso del ejercicio que nos ocupa consiste en comprobar cuáles cumplen la propiedad considerada y cuáles no.

5.1.3.1. Generación del enunciado

Para la generación del enunciado se seguirán los siguientes pasos:

1. **Elegir el número de nodos de los que se compone el grafo que vamos a considerar y elegir el número de enlaces o aristas entre dichos nodos.** Al igual que en los ejercicios anteriores, este número no puede ser muy grande por el contexto en el que estamos trabajando, una pregunta de test de un examen o ejercicio de evaluación. Un número razonable, podría ser un valor comprendido entre 4 y 6 ya que un valor superior a 6 haría el grafo demasiado grande, un valor inferior a 2 nos impide plantear el problema, ya que necesitamos más de un enlace en el grafo G. Los nodos los representamos con las primeras letras mayúsculas del abecedario $A, B, C, D, E, ...$ Los enlaces quedan representados según el tipo de grafo:

   a) grafos no dirigidos acíclicos (GNA) por un guión entre los nodos que constituyen el enlace, por ejemplo para representar el enlace entre A y B escribimos $A - B$.

   b) grafos dirigidos acíclicos (GDA) quedan representados por una flecha dirigida entre los nodos que constituyen el enlace, por ejemplo para representar el enlace desde A hasta B escribimos $A \to B$.

2. Una vez elegido el número de nodos y de enlaces de los que se compone nuestro grafo, en este punto nos centraremos en implementar un **generador de enlaces**. El

número máximo de enlaces que podemos generar vendrá dado por la combinación de m elementos tomados de n en n. Son los grupos que podemos hacer de entre m elementos tomados de n en n diferenciándose, un grupo de otro, en tener algún elemento distinto. En función de que el grafo sea o no dirigido tendremos una fórmula distinta para calcular el número de enlaces:

a) Para el caso de los grafos no dirigidos (GNA):

$$C_m^n = \frac{m!}{n!(m-n)!}$$

b) Para el caso de los grafos dirigidos (GDA), habrá que tener también en cuenta el sentido de los mismos por los que el número máximo vendrá dado por la fórmula:

$$C_m^n = \frac{m!}{n!(m-n)!} \times 2$$

3. Realizar un programa que genere las distintas combinaciones teniendo en cuenta la fórmula correspondiente anterior. Una vez generadas todas las combinaciones elegiremos entre ellas hasta completar el número de enlaces deseado. Cada una de estas combinaciones representa una arista o enlace dirigido o no dirigido que se establece entre dos nodos. En este punto ya tendríamos generado el grafo que vamos a considerar en el enunciado.

4. Una vez generada el grafo, se pasa a **generar las combinaciones entre las distintos nodos consideradas** para plantear las relaciones, en este caso de independencia probabilista entre dos nodos $I_p(X, Y)$. Estas relaciones las elegiremos aleatoriamente de las combinaciones generadas en el punto anterior. Para plantear las relaciones de independencia probabilista condicional $I_p(X, Y \mid Z)$, deberemos generar todas las posibles combinaciones de 2 variables respecto de una tercera variable o conjunto de variables. De todas esas combinaciones generadas, elegiremos las que se vayan a plantear en el enunciado.

Con todo, a partir de este momento, disponemos de los elementos necesarios para poder plantear una pregunta de relación independencia probabilística o independencia probabilística condicional entre los nodos o variables de un grafo. En el apartado siguiente estudiaremos como plantear las posibles respuestas que se plantean a la pregunta expuesta en el enunciado y cuál es la respuesta correcta.

5.1.3.2. Generación de la Solución

Una vez generado el enunciado de la pregunta, el programa ha de ser capaz de dar con la solución del mismo. Tendremos que determinar para cada apartado si la relación es verdadera o falsa, es decir si se cumple o no la relación de independencia probabilística o independencia probabilística condicional para las variables consideradas en cada apartado. O lo que es lo mismo determinar si los nodos están o no conectados teniendo o no en cuenta una tercera variable. Ver el apartado de separación en grafos en el capítulo de fundamentos.

- Para comprobar si la relación $I_p(X, Y)$ es verdadera o falsa tendremos que recurrir a la representación del grafo y determinar todos los caminos activos entre los dos nodos considerados en la expresión X e Y . Si existe al menos un camino activo entre las dos variables consideradas concluiremos que la relación no se cumple pues las variables X e Y no están separadas en el grafo o no son independientes, las variables X e Y están por tanto conectadas (se cumple $\neg I_p(X, Y)$). Si no hay caminos activos entre ellas, la relación $I_p(X, Y)$ será cierta.

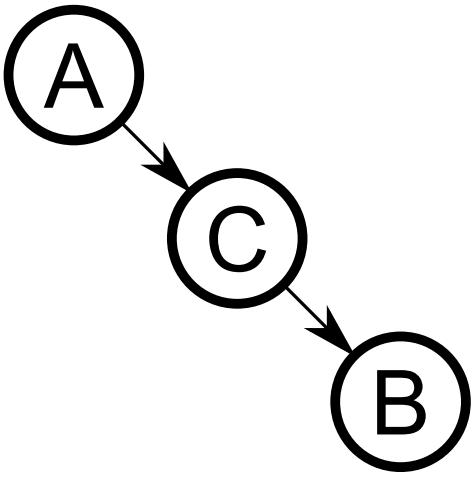
- Del mismo modo para los apartados de relación de independencia condicional $I_p(X, Y \mid Z)$ recurriremos al grafo para determinar los caminos activos entre las dos variables a la izquierda de la barra. Si encontramos algún camino activo entre las dos variables, concluiremos que la relación no se cumple $\neg I_p(X, Y \mid Z)$. Si los caminos que estaban activos han quedado bloqueados por la variable o variables a la derecha de la barra concluiremos que la relación se cumple $I_p(X, Y \mid Z)$.

En el caso de que el grafo sea no dirigido, el problema consistirá en buscar todos los posibles caminos que existen entre un par de nodos dados. Una vez obtenidos dichos caminos analizamos si están o no activos. en el caso de una relación entre dos variables el hecho de encontrar un camino ya indica que está activo. Cuando la relación está condicionada a una tercera variable o conjunto de variables habrá que estudiar si los caminos activos entre las dos variables del lado izquierdo de la barra han quedado todos ellos bloqueados o no, también si existe algún camino que no pase por las variables consideradas en la parte derecha de la barra.

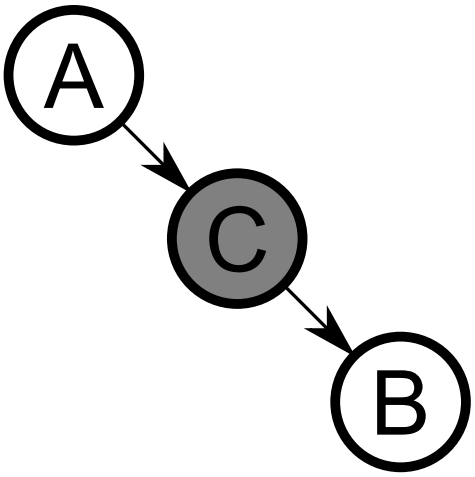
En el caso de los grafos dirigidos la situación es un poco más complicada, el problema consistirá en buscar todos los posibles caminos que existen entre un par de nodos dados. Una vez obtenidos dichos caminos analizamos si están o no activos:

- en el caso de una relación entre dos variables el hecho de encontrar un camino (en cualquier sentido) ya indica que está activo.

- Cuando la relación está condicionada a una tercera variable o conjunto de variables habrá que estudiar si los caminos activos entre las dos variables del lado izquierdo de la barra han quedado todos ellos bloqueados o no, también si existe algún camino que no pase por las variables consideradas en la parte derecha de la barra. Para este caso se pueden dar las tres situaciones de la figura 5.3

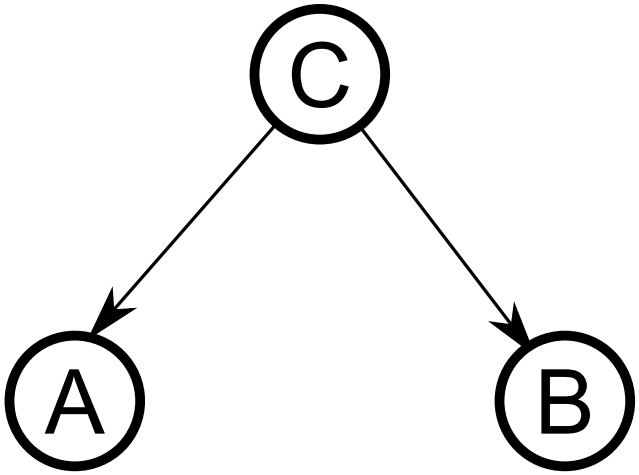


(a) Camino activo: $\neg I_G(A, B)$.

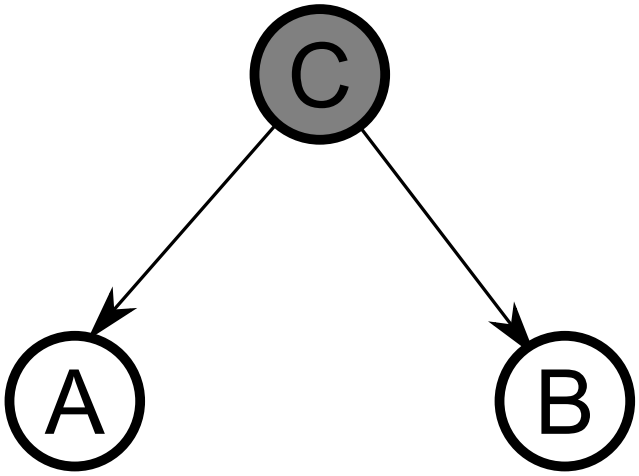


(b) Camino bloqueado: $I_G(A, B|C)$.

Figura 5.3: Cabeza-cola.

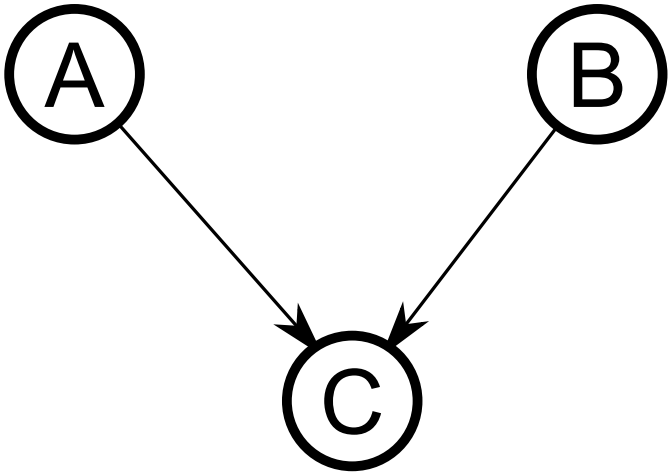


(a) Camino activo: $\neg I_G(A, B)$.

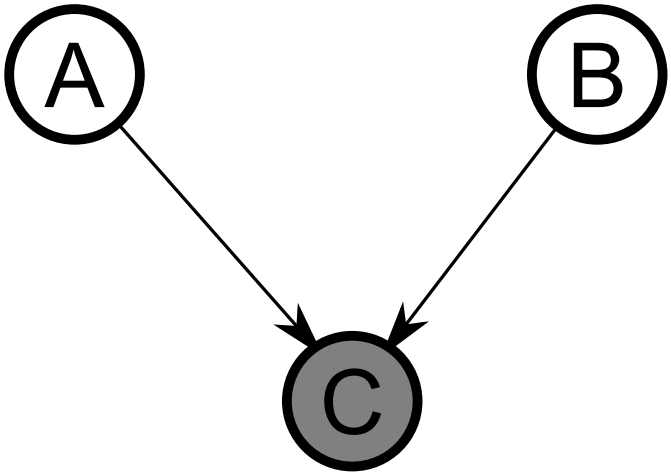


(b) Camino bloqueado: $I_G(A, B|C)$.

Figura 5.4: Cola-cola.



(a) Camino activo: $I_G(A, B)$.



(b) Camino bloqueado: $\neg I_G(A, B|C)$.

Figura 5.5: Cabeza-cabeza.

Teniendo en cuenta cada una de las situaciones presentadas en las figuras anteriores, podremos determinar si una determinada relación establecida entre los nodos o variables aleatorias de un grafo dirigido se cumple o no. Del mismo modo la información de refuerzo la obtendremos calculando los caminos activos entre un par de nodos tanto en el caso de grafos no dirigidos como de grafos dirigidos.

Como conclusión de esta sección, hemos analizado el problema a resolver y especificado los pasos que se han de seguir en el diseño de los algoritmos que nos permitan determinar si se cumplen o no las relaciones de independencia condicional entre variables aleatorias de un sistema tanto en el caso de los grafos. La figura 5.6 representa el diagrama de componentes software para el desarrollo de este segundo bloque temático de preguntas.

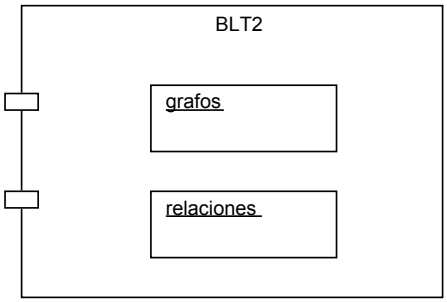


Figura 5.6: Componente que define una pregunta del bloque temático 2.

5.1.4. Inferencia en Redes Bayesianas

En este apartado analizaremos la inferencia en redes bayesianas. Los apartados anteriores constituyen la base de este tercer bloque temático. La primera parte (5.1.2), nos permite obtener la parte cuantitativa de una red bayesiana, mientras que el segundo bloque (5.1.3) nos sirve para obtener la estructura o la representación gráfica de la red bayesiana, es decir la parte cualitativa. Para realizar la inferencia sobre una red bayesiana nos apoyaremos en el programa de software libre desarrollado en la UNED *OpenMarkov* [1] para el estudio de modelos gráficos probabilistas. También deberemos construir un algoritmo que nos sirva para la generación aleatoria de redes bayesianas (Ide et al.,).

5.1.4.1. Generación del enunciado

El enunciado que se plantea para este tipo de preguntas, consistirá en la presentación de una red bayesiana, del grafo que representa su estructura, acompañado de la parte cuantitativa de la misma, es decir de las tablas de probabilidad condicionada (CTP por sus siglas en inglés) de cada una de las variables o nodos que configuran la red. Además se presenta, para cada variable, el conjunto de los distintos estados o valores discretos que puede tomar cada una de las mismas. A partir de dicha información, se plantea como

[1] Haremos uso de la librería de *OpenMarkov* en modo API para la inferencia sobre redes bayesianas.

parte del enunciado la introducción en la red de uno o varios hallazgos sobre alguna o algunas variables o nodos de la red y la pregunta que se plantea consiste en indicar cual es el valor predictivo (positivo o negativo para el caso de variables aleatorias binarias) de alguna variable de interés de la red, es decir de la probabilidad a posteriori de algún valor de los que puede tomar la variable aleatoria. En este bloque temático vamos a considerar tres tipos distintos o variantes de preguntas que son las siguientes:

- pregunta de tipo 7.- se presenta como red bayesiana objeto de estudio la red *Disease-Test*. Esta red está formada por dos nodos y un enlace. Los nodos son los correspondientes a una determinada enfermedad (X) y el test que se realiza para confirmar o descartar la presencia de dicha enfermedad (Y) en una determinada población. El enlace va desde el nodo enfermedad al nodo test. El enunciado da como datos de entrada los siguientes ver 4.3.1:

  - la **prevalencia** de la enfermedad en la población en forma de porcentaje o proporción, es decir la probabilidad a priori de padecer la enfermedad: $P\left(+x\right)$.
  - La **sensibilidad** del test, es decir la probabilidad de que el test de positivo, sabiendo que se tiene la enfermedad: $P(+y|+x)$.
  - Por último la **especificidad**, es decir la probabilidad de que el resultado del test sea negativo sabiendo que la enfermedad no está presente: $P\left(\neg y \mid \neg x\right)$.

A partir de estos datos, la pregunta que se plantea es calcular los valores predictivos del test para la enfermedad $Y(+y, \neg y)$ el valor predictivo positivo $P\left(+x \mid +y\right)$ o la probabilidad de padecer la enfermedad cuando el resultado del test ha dado positivo así como el valor predictivo negativo $P\left(\neg x \mid \neg y\right)$ o probabilidad de descartar la presencia de la enfermedad cuando se ha obtenido resultado negativo en el test.

Como segunda parte de esta pregunta se plantea la introducción del hallazgo obtenido como resultado de aplicar el test de la enfermedad y se pregunta por el valor predictivo positivo del test, es decir por la probabilidad de padecer la enfermedad sabiendo que el resultado del test ha dado positivo, es decir $P(+x|+y)$.

- pregunta de tipo 8.- en este caso se plantea como objeto de estudio una de los tres tipos de redes conocidos y que vienen incluidas en la factoría proporcionada por el programa *OpenMarkov*. El enunciado presenta una de las tres redes bayesianas siguientes: red *bN_ABC*, red *bN_XYZ* o la red *Asia*. Se especifica en el enunciado, al igual que en el caso anterior la estructura de la red o parte cualitativa junto con los distintos valores o estados que puede tomar cada una de las variables que

la forman, acompañada de su parte cuantitativa. Tras introducir una determinada evidencia de manera aleatoria en la red se pregunta por la probabilidad a posteriori de algún valor de una variable de interés elegida también al azar.

- pregunta de tipo 9.- En este tipo de preguntas se plantea el mismo caso anterior pero en este caso se parte de una red bayesiana generada aleatoriamente. Para ello habrá que especificar el número de nodos de los que se compone la red. Este número estará comprendido entre dos y ocho, que es el intervalo que se considera apropiado para este tipo de preguntas de test sobre inferencia en redes bayesianas. Igualmente la pregunta que se plantea consiste en averiguar la probabilidad a posteriori de alguno de los valores (elegido aleatoriamente) de la variable de interés (elegida también aleatoriamente), a partir de la red bayesiana dada, tanto de su estructura como de su parte cuantitativa.

5.1.4.2. Generación de la Solución

La generación de la solución en los tres casos que se plantean en el apartado anterior, pasará por hacer uso de la librería de generación de modelos gráficos probabilísticos *OpenMarkov* comentada al inicio de este apartado. Este software de libre distribución desarrollado en la UNED, nos servirá para obtener la distribución de probabilidad conjunta asociada a la red bayesiana y que obtenemos a partir de su factorización donde la probabilidad de cada nodo puede escribirse como la factorización de sus antecesores (ver4.3.1). Con esta tabla, de probabilidad condicionada de la red, y las correspondientes proyecciones que se plantean en el enunciado de la pregunta podemos obtener una forma de determinar el valor de la probabilidad a posteriori buscada en el enunciado, del valor que toma la variable de interés.

El valor vendrá dado por la siguiente expresión:

$$P(valorBuscado) = \frac{Proyecc(valorBuscado,evidencia)}{Proyecc(evidencia)}$$

de donde el numerador vendrá dado por la proyección sobre la tabla de probabilidad condicionada obtenida a partir de *OpenMarkov* de la probabilidad conjunta del valor buscado y la evidencia introducida mientras que el denominador vendrá dado por la proyección de la evidencia introducida.

Por ejemplo, ante el planteamiento:

<<Si se introduce en la variable B el hallazgo "*absent*". ¿Cuál es la probabilidad asociada al valor "*absent*" de la variable A?>>

tendríamos la siguientes expresión:

$$P(\neg a) = \frac{proyección(\neg b, \neg a)}{proyección(\neg b)}$$

El valor de dicho cociente coincidirá con el valor buscado. Esta es una forma sencilla de obtener la solución, que nos permitirá obtener el refuerzo para cada tipo de pregunta planteada en los apartados anteriores.

Una forma un tanto más compleja de obtener el mismo resultado sería mediante la aplicación del teorema de Bayes, de su forma normalizada (ver apartado 4.3.1).

Dada la complejidad de los cálculos al aplicar directamente el teorema de Bayes en los cálculos de la probabilidad a posteriori optamos por el método explicado para obtener la solución y el refuerzo a las preguntas relacionadas con inferencia en redes bayesianas.

La figura 5.7 representa el diagrama de componentes correspondiente a esta parte del desarrollo.

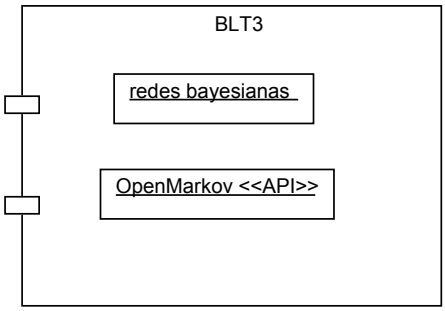


Figura 5.7: Componente que define una pregunta del bloque temático 3.

5.2. Análisis de los requisitos de usuario

Para el análisis de los requisitos funcionales del usuario vamos a hacer uso de los diagramas UML. En concreto vamos a expresar la funcionalidad de nuestro sistema mediante los diagramas de casos de uso. Como se puede leer en la wikipedia, un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra

la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones.

En los siguientes apartados vamos a describir los distintos diagramas de casos de uso considerados en este proyecto.

5.2.1. Actores

Los actores son los usuarios del sistema que se está modelando. Cada actor tendrá un rol bien definido, y en el contexto de ese rol tendrá interacciones útiles con el sistema. Una persona puede desempeñar el papel de más de un actor, a pesar de que sólo asumirá un papel para una interacción de un caso de uso. Es el caso del autor de este proyecto que adopta el papel de "*Colaborador*" en la creación de la librería, de "*Profesor*" para dar de alta la asignatura en el sistema *Siette* y definir los distintos tipos de preguntas y de "*Alumno*" a la hora de realizar las pruebas del sistema.

En este apartado vamos a analizar los distintos tipos de usuario o perfiles que hacen uso del sistema y que van a constituir los actores de los casos de uso (ver figura 5.8). Vamos a realizar una clasificación de los actores entre abstractos y concretos. En los siguientes apartados consideraremos cada uno de ellos.

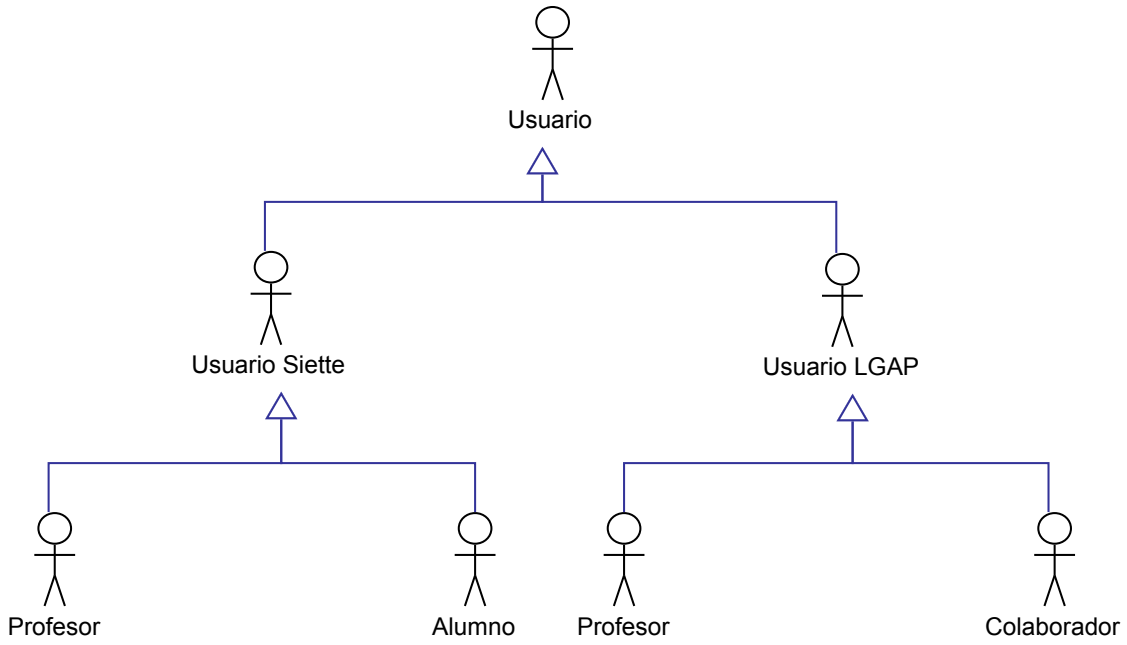


Figura 5.8: Dependencias entre los diferentes actores participantes en el sistema.

5.2.1.1. Actores abstractos

En nuestro sistema vamos a considerar tres actores abstractos que nos van a servir para clasificar los distintos tipos de usuarios. Consideraremos como sistema, tanto la interacción con el sistema *Siette* como con la librería de generación automática de preguntas a través de un programa cliente java. Partimos del actor abstracto *Usuario* del que derivan los actores *UsuarioSiette* y *UsuarioLGAP*.

- El actor *UsuarioSiette* representa el usuario del sistema *Siette* una vez que se ha integrado en el mismo la librería *LGAP* desarrollada en este proyecto.

- El actor *UsuarioLGAP* representa el usuario que hace uso de la funcionalidad proporcionada por la librería a través de los siguientes interfaces según el tipo de actor:

  - Programa cliente desarrollado para la generación de exámenes de test en formato PDF. Será usado por el actor derivado "Profesor" para la elaboración de exámenes marcando o no la solución a cada pregunta.
  - Entorno integrado de desarrollo tipo Eclipse que será utilizado por el actor derivado "Colaborador" para editar los programas fuentes que forman parte de la librería con el fin de corregir o mejorar alguna característica de la generación automática de preguntas. Asimismo podrá incorporar a la librería nuevos fuentes que se correspondan con la implementación de nuevos tipos de preguntas relacionadas con la misma asignatura o con otras que se puedan plantear.

5.2.1.2. Actores concretos

Profesor Consideraremos el actor Profesor en dos situaciones distintas:

- Dentro del sistema *Siette*, el perfil del profesor será el que confecciona cada pregunta a partir del repertorio de tipos de pregunta que ofrece la librería desarrollada. Además, definirá los test con las preguntas disponibles para ser realizados por los alumnos y usuarios de *Siette* (Siette, 2013).

- Por otro lado el profesor hará uso del programa cliente de la librería que permite confeccionar exámenes de test en formato PDF para ser resueltos por los alumnos y genera los exámenes con las soluciones para proceder a la corrección de los mismos una vez han sido realizados por los alumnos.

Alumno El perfil del alumno se corresponde con un estudiante que hace uso del sistema *Siette* y emplea las ayudas y la documentación como base teórica del aprendizaje. Tiene o está adquiriendo un conocimiento de la base teórica de la asignatura Métodos Gráficos Probabilistas y ejecuta los test que se han definido en el sistema *Siette* para dicha asignatura para conseguir una formación práctica. Por otro lado resolverá exámenes proporcionados por el profesor en formato PDF y que han sido generados por un programa cliente de la librería desarrollada de generación de preguntas. En este caso el alumno no interacciona con ningún sistema sino que recibe el documento impreso a partir del PDF generado por el profesor en su interacción con el programa cliente de la librería *LGAP*. El alumno resuelve su examen y se lo entrega al profesor para su corrección.

Colaborador El tercer perfil concreto que vamos a considerar es el de colaborador en el desarrollo de nuevos tipos de preguntas, se corresponde con un estudiante con una base de conocimientos suficiente para aportar nuevos contenidos teóricos al sistema. Este mismo perfil colaborará en la mejora de las preguntas o en el desarrollo de nuevos tipos de preguntas relacionadas con la asignatura o de otras que se puedan proponer para el desarrollo de otras librerías similar a la implementada.

5.2.2. Casos de uso

El modelo de casos de uso es un catálogo de la funcionalidad del sistema descrito mediante el lenguaje UML. Cada caso de uso representa una única interacción, repetible que un usuario o "actor" experimenta utilizando el sistema. Un caso de uso general, incluye uno o varios "escenarios" que describen las interacciones que van entre el Actor y el Sistema, y documentan los resultados y las excepciones que se producen desde la perspectiva del usuario. Los casos de uso pueden incluir a su vez otros casos de uso como parte de un patrón más general de la interacción y también pueden ser extendidos por otros casos de uso para manejar condiciones excepcionales. Cada interacción puede ser especificada utilizando escenarios, diagramas de secuencia, diagramas de comunicación y otros diagramas dinámicos o descripciones textuales que en su conjunto describen el funcionamiento del sistema cuando se ve como una caja negra que interactúa con el usuario.

En este apartado vamos a establecer los distintos casos de uso, es decir definiremos como se va a hacer uso del sistema por parte de los distintos actores identificados en el apartado anterior.

5.2.2.1. Casos de uso de la librería *LGAP* integrada en *Siette*

En el primer diagrama vamos a considerar el sistema formado por la herramienta *Siette* en el que ya se ha incorporado la librería de generación automática de preguntas (*LGAP*) implementada en este proyecto. Una vez se desarrollada la librería, esta pondrá a disposición del sistema *Siette* un repertorio de preguntas relacionadas con la asignatura Métodos Gráficos Probabilísticos. En la figura 5.9 representamos las distintas acciones que se pueden llevar a cabo en este sistema por los distintos actores. Los actores que intervienen en este caso son los usuarios del sistema *Siette* (derivan de los actores abstractos *Usuario*/*UsuarioSiette*) que pueden ser de dos tipos: profesor o alumno.

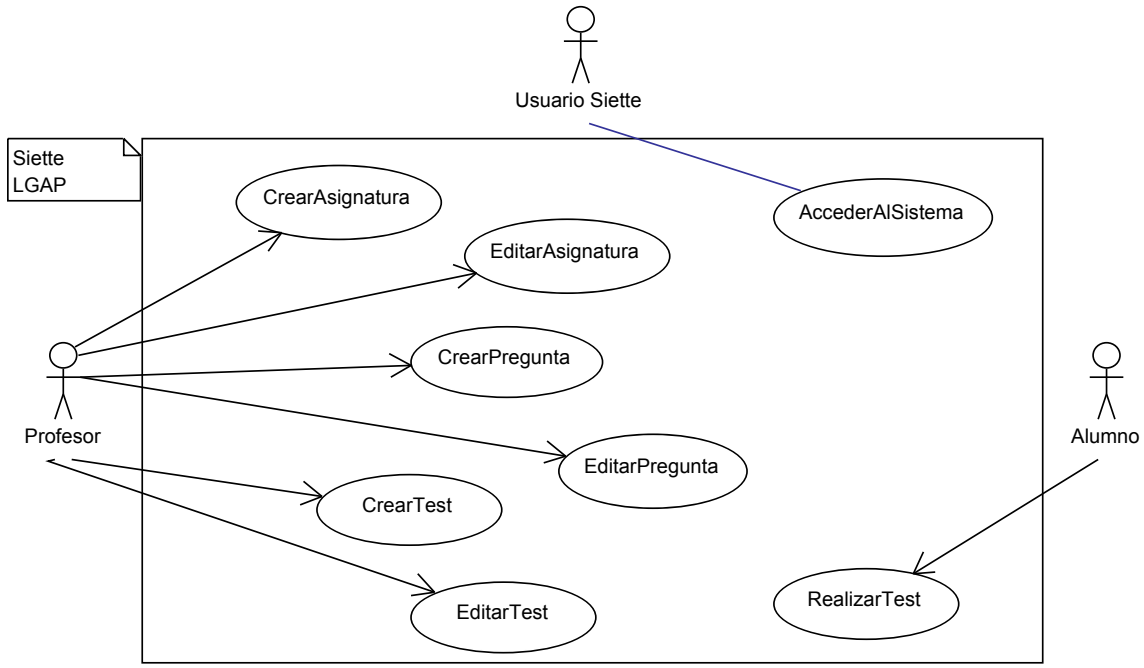


Figura 5.9: Diagrama de casos de uso de la librería *LGAP* integrada en *Siette*.

A continuación comentamos cada uno de los casos de uso que aparecen en el diagrama:

- El caso de uso *AccederAlSistema* lo utiliza cualquier tipo de usuario del sistema *Siette*. Representa la acción de logarse al sistema tanto para usuarios con el rol de profesor como para usuarios con el rol de alumno. Por eso hemos representado con el actor "*UsuarioSiette*". En la tabla 5.1, para cada uno se especifica su nombre y el actor que interviene en el caso de uso. Se especifican los pasos de la ejecución normal del caso de uso y cuando sea necesario las posibles alternativas que pueden aparecer como posibles errores o datos mal introducidos.

Tabla 5.1: Caso de uso *AccederAlSistema.*

Caso de Uso: *AccederAlSistema*	
Actor: Usuario Siette (Profesor, Alumno)	
Curso normal	**Alternativas**
1) El usuario introduce en su navegador la url para acceder al sistema: www.siette.org	
2) Aparece la pantalla de entrada al sistema	
3) Usuario introduce nombre y contraseña	
4) Sistema comprueba datos introducidos	4.1) Si no autorizado, mensaje error
5) Sistema muestra pantalla asignaturas	

- El siguiente caso de uso es el de crear una asignatura, llevado a cabo por el profesor. La asignatura que se creará será "Métodos Gráficos Probabilistas" con tres subtemas:

  - Relaciones de Independencia Probabilística sobre una distribución de probabilidad (parte cuantitativa de una red bayesiana).
  - Relaciones de Independencia Probabilística sobre un grafo (parte cualitativa de la red bayesiana).
  - Inferencia en Redes Bayesianas.

En la tabla 5.2 representamos el caso de uso correspondiente.

Tabla 5.2: Caso de uso *CrearAsignatura.*

Caso de Uso: *CrearAsignatura*	
Actor: Profesor	
Curso normal	**Alternativas**
1) Profesor pulsa botón "Nueva Asignatura"	
2) Profesor introduce datos mínimos nueva asignatura	
3) Profesor edita todas las pestañas, completa la información necesaria, pulsa "Guardar cambios"	3.1) Si se produce un error en la edición, avisa error producido para su corrección.

- En el caso de uso siguiente 5.3, representamos la acción de editar asignatura una vez creada.

Tabla 5.3: Caso de uso *EditarAsignatura.*

Caso de Uso: *EditarAsignatura*	
Actor: Profesor	
Curso normal	**Alternativas**
1) El profesor pulsa el botón "Editar Asignatura"	
2) Selecciona la asignatura a editar	
3) El profesor edita todas las pestañas con la información necesaria, pulsa "Guardar cambios"	3.1) Si se produce error en la edición, avisar para su corrección.

- A continuación, el caso de uso *CrearPregunta* (5.4) nos permite crear una pregunta de alguno de los tipos ofrecidos por el repertorio de la librería LGAP. En el apartado de anexos podemos encontrar una descripción completa de los distintos tipos de preguntas disponibles y de la parametrización ofrecida para definir cada pregunta.

Tabla 5.4: Caso de uso *CrearPregunta.*

Caso de Uso: *CrearPregunta*	
Actor: Profesor	
Curso normal	**Alternativas**
1) El profesor pulsa el botón "Editar Asignatura"	
2) El profesor pulsa el botón "Nuevo"	
3) El profesor pulsa el botón "Nuevo Pregunta"	
4) Elige tipo pregunta de las opciones mostradas	
5) Edita la información mínima y pulsa el botón "Insertar"	5.1) Si se produce un error en la edición, avisa del error producido para su corrección.
6) Edita el resto de pestañas con la información de la asignatura y pulsa el botón "Guardar cambios"	6.1) Si se produce un error en la edición, avisa del error producido para su corrección.

- El siguiente caso de uso (5.5) representa la edición de una pregunta de la asignatura una vez creada:

Tabla 5.5: Caso de uso *EditarPregunta.*

Caso de Uso: *EditarPregunta*	
Actor: Profesor	
Curso normal	**Alternativas**
1) El profesor pulsa el botón "Editar Asignatura"	
2) Elige la asignatura a editar	
3) Elige la pregunta a editar dentro de la asignatura	
6) Edita resto de pestañas con toda la información de la asignatura y pulsa "Guardar cambios"	6.1) Si se produce un error en la edición, avisa del error producido para su corrección.

- Una vez definida la asignatura y todas las preguntas posibles para esa asignatura pasamos a definir los test que serán realizados por los alumnos, el caso de uso *CrearTest* queda especificado en la tabla 5.6.

Tabla 5.6: Caso de uso *CrearTest.*

Caso de Uso: *CrearTest*	
Actor: Profesor	
Curso normal	**Alternativas**
1) El profesor pulsa "Editar Asignatura"	
2) Selecciona la asignatura de la lista	
3) Pulsa el botón Test	
4) Pulsa el botón "Nuevo test"	
5) Rellena información mínima y pulsa "Insertar"	5.1) Si se produce un error en la edición, avisa error producido para su corrección.
6) Informa resto de información en las pestañas y pulsa "Guardar cambios"	6.1) Si se produce un error en la edición, avisa error producido para su corrección.

- En *EditarTest* (5.7) editamos el test una vez creado para modificar alguna característica del mismo:

Tabla 5.7: Caso de uso *EditarTest.*

Caso de Uso: *EditarTest*	
Actor: Profesor	
Curso normal	**Alternativas**
1) El profesor pulsa "Editar Asignatura"	
2) Selecciona la asignatura de la lista	
3) Pulsa el botón Test	
4) Selecciona el test a editar	
5) Rellena información en pestañas y pulsa "Guardar cambios"	5.1) Si se produce un error en la edición, avisa error producido para su corrección.

El caso de uso realizar test (5.8) es llevado a cabo por un alumno que ingresa en el sistema *Siette* para evaluar sus conocimientos sobre la asignatura Métodos Gráficos Probabilistas. El alumno utiliza el sistema como una herramienta más a su disposición para guiar y enriquecer su propio proceso de aprendizaje de la asignatura en cuestión. En la tabla siguiente representamos el caso de uso.

Tabla 5.8: Caso de uso *RealizarTest.*

Caso de Uso: *RealizarTest*	
Actor: Alumno	
Curso normal	**Alternativas**
1) Alumno ingresa en sistema (ver c.u *AccederSistema*)	
2) Pulsa el botón "Hacer un test"	
3) Selecciona la asignatura	
4) Selecciona el test a realizar	
5) Pulsa el botón "Empezar"	
6) Sistema mostrando cada pregunta y alumno contesta y pulsa sobre siguiente Pregunta	6.1) Si se falla una pregunta se muestra el refuerzo.

5.2.2.2. Casos de uso del programa cliente de la librería *LGAP*

En este segundo diagrama vamos a representar los casos de uso del sistema consistente en la propia librería *LGAP* que proporciona un repertorio de preguntas que nos permitirán generar exámenes con preguntas de test en formato PDF. El interfaz de acceso al sistema, en este caso la librería, varía en función del tipo de usuario que accede al sistema. Así en el caso del profesor interaccionará con la librería a partir del programa cliente desarrollado que hace uso de la misma y le ofrece la funcionalidad de crear exámenes de test en formato

PDF. Para el caso del colaborador accede al sistema a partir, por ejemplo, de un entorno integrado de desarrollo. Los actores en este caso podrán ser de dos tipos:

- el "*Profesor*" que hace uso del sistema para generar un examen de test en formato PDF para que sea resuelto en papel por los alumnos. Una vez resueltos los exámenes, el profesor deberá corregirlos. Para ello hará uso de nuevo del programa cliente para generar el mismo examen con las soluciones a cada pregunta propuesta.

- por otro lado tendremos al actor "*Colaborador*" que editará la librería desarrollada para mejorar o corregir alguna característica o bien para incorporar al sistema nuevos tipos de preguntas al repertorio ofrecido para ampliar el temario de la asignatura o incorporar nuevas preguntas que permitan definir otra asignatura.

En la figura 5.10 representamos ambas situaciones:

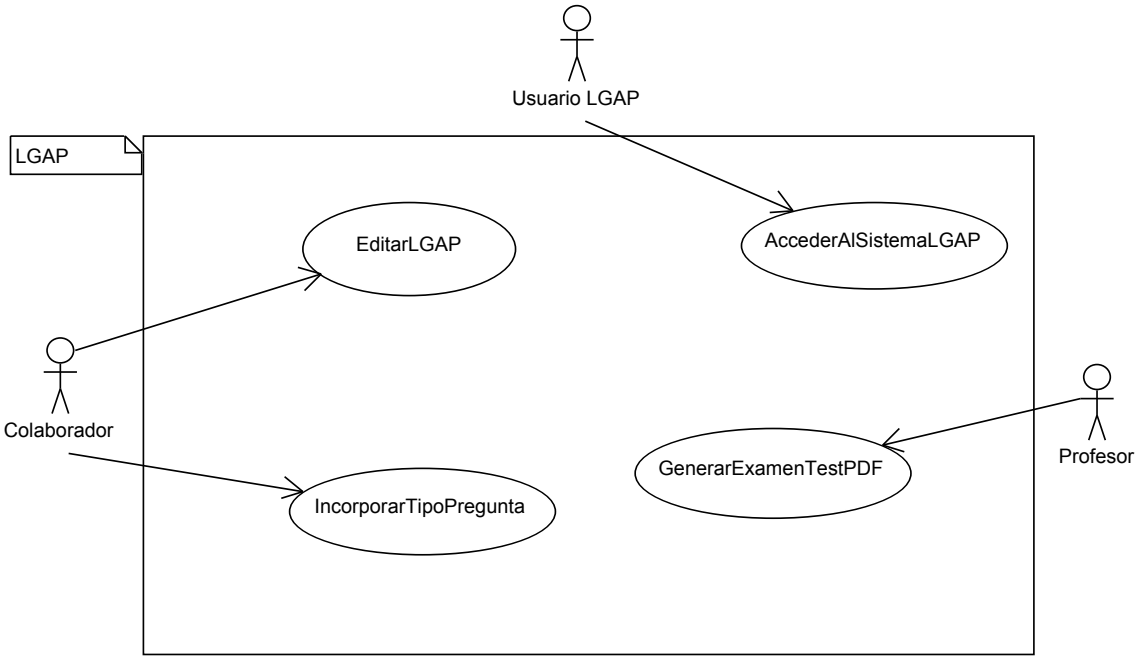


Figura 5.10: Diagrama de casos de uso de la librería *LGAP*.

A continuación vamos a describir cada uno de los casos de uso detalladamente. En cada tabla especificaremos el nombre de cada uno y el actor que interviene en cada caso. Se especifican los pasos de la ejecución normal del caso de uso y cuando sea necesario las posibles alternativas que pueden aparecer como posibles errores o datos mal introducidos.

- El caso de uso "*AccederAlSistemaLGAP*" 5.9 representa el acceso a la librería de generación automática de preguntas LGAP de las dos formas posibles ya comenta-

das: mediante el programa cliente o desde un entorno integrado de desarrollo como puede ser Eclipse.

Tabla 5.9: Caso de uso *AccederAlSistemaLGAP.*

Caso de Uso: *AccederAlSistemaLGAP*	
Actor: Usuario LGAP (Profesor, Colaborador)	
Curso normal	**Alternativas**
1) El usuario ejecuta el programa cliente.	1.1) Usuario Ejecuta el *IDE*.

- En el caso de uso que viene a continuación 5.10, el actor "Profesor" interactúa con el programa cliente para generar un examen en formato PDF que posteriormente entregará a sus alumnos para su realización:

Tabla 5.10: Caso de uso *GenerarExamenTestPDF.*

Caso de Uso: *GenerarExamenTestPDF*	
Actor: Profesor	
Curso normal	**Alternativas**
1) El usuario ejecuta el programa cliente	
2) Elige la opción de generar examen PDF	
3) El programa pide nº de preguntas del examen	3.1) Si se marca la opción "Con solución", respuesta correcta a cada pregunta.
4) Se genera test PDF solicitado.	

- En el siguiente caso de uso 5.11 representamos la acción de editar la librería de generación automática de preguntas por parte de un usuario "Colaborador" que pretende modificar alguna característica de alguna pregunta, corregir algún error o realizar alguna variación de alguna de las preguntas del repertorio ofrecido por la librería.

- El siguiente caso de uso 5.12 representa el hecho de ampliar el sistema introduciendo nuevos tipos de pregunta por parte de un usuario colaborador. Este colaborador amplia el desarrollo realizado para incorporar nuevos tipos de preguntas de la misma asignatura o de otras.

Tabla 5.11: Caso de uso *EditarLGAP.*

Caso de Uso: *EditarLGAP*	
Actor: Colaborador	
Curso normal	**Alternativas**
1) El usuario entra en el IDE	
2) Edita fuentes para modificar tipo de pregunta	
3) El usuario genera nuevo .jar con la(s) pregunta(s) modificada(s)	3.1) Si se producen errores de compilación, etc se corrigen.
4) Se integra la nueva versión de la librería en Siette	

Tabla 5.12: Caso de uso *IncorporarTipoPregunta.*

Caso de Uso: *IncorporarTipoPregunta*	
Actor: Colaborador	
Curso normal	**Alternativas**
1) El usuario entra en el IDE	
2) Edita nuevos fuentes para definir nuevo tipo de pregunta	
3) El usuario genera nuevo .jar con nuevas preguntas disponibles.	3.1) Si se producen errores de compilación, etc se corrigen.
4) Se integra la nueva versión de la librería en Siette	

A partir de aquí pondríamos a disposición de un usuario profesor de Siette un nuevo tipo de pregunta a partir de la cuál podría plantear nuevas preguntas Siette de la asignatura que posteriormente incorporaría a los test.

Por su parte, el programa cliente no sufriría cambios ya que se limita a solicitar al usuario el número de preguntas de los que se compondrá el examen a generar y el programa elige las preguntas solicitadas a partir de las preguntas ofertadas por el repertorio de la librería de manera aleatoria para generar el examen de test.

Capítulo 6

Diseño

En el presente capítulo vamos a realizar el diseño de la solución para el desarrollo de la librería de generación automática de preguntas (*LGAP*) que constituye el objetivo principal del presente proyecto de fin de carrera. En el capitulo anterior, hemos realizado un análisis de los distintos tipos de pregunta que vamos a plantear en la librería de generación automática de preguntas identificando la parte común y definiendo los tres bloques temáticos de preguntas que se van a tratar en el dominio de aplicación de la materia teórica sobre las que versarán cada una de las preguntas. Del mismo modo hemos realizado un análisis de los requisitos funcionales identificados en el proyecto describiendo la funcionalidad del mismo a partir de los diagramas de casos de uso en UML.

En la realización de nuestro diseño vamos hacer uso de nuevo del lenguaje de modelado de sistemas UML (por sus siglas en inglés, *Unified Modeling Language*) (C.Martin, 2004). Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. De todos los diagramas disponibles en la especificación del lenguaje, utilizaremos principalmente dos tipos: diagramas de componentes y diagrama de clases.

6.1. Componentes comunes

En el capítulo anterior hemos detallado todos los casos de uso que vamos a tener en nuestro sistema. En este apartado nos centraremos en las tareas de diseño que tienen que ver con los componentes comunes que van a dar soporte a la implementación de los componentes de la librería. El componente principal será un tipo de pregunta que responde a un formato bien definido. A partir de este formato definido crearemos distintas instancias

de la pregunta que darán lugar a la generación automática de cada tipo de pregunta donde identificamos una parte fija y una parte variable que se generará de manera automática y que dará lugar a las distintas instancias u ocurrencias de cada uno de los formatos o tipos de preguntas. Dejaremos para el apartado 6.2 los distintos tipos de preguntas que se van a considerar. Analizamos primeramente la parte común que dará soporte al resto de la implementación. Señalaremos en el apartado que corresponda la tarea de análisis identificada en el plan de proyecto (ver sección 3.3).

6.1.1. Excepciones

En la librería desarrollada definiremos un sistema de control de excepciones propio. Esto nos permitirá identificar con mayor claridad los posibles problemas que se puedan producir en la generación de cada tipo de pregunta y proporcionar al usuario final de la librería la información más detallada posible sobre las posibles circunstancias de error que se pueden dar con el uso de la misma.

Para ello crearemos un paquete Java en el que se recogen los distintos tipos de excepciones que se pueden dar y la relación jerárquica que se establece entre las distintas clases de excepciones. La figura 6.1 representa la jerarquía de clases definida para el paquete excepciones. En ella observamos una primera clase genérica llamada *LgapException* que hereda de la clase *Exception* de Java. Esta clase se especializa en dos tipos de excepciones:

- *LgapRelipException* para tratar las excepciones relativas a las relaciones de independencia probabilística entre variables aleatorias, respecto de una distribución de probabilidad *(LgapSystemException)* o en un grafo *(LgapGrafosException)* .
- *LgapRBException* para excepciones que se producen a la hora de tratar la inferencia en redes bayesianas.

6.1.2. Integración *LGAP* con *Siette*

En primer lugar presentamos el diagrama 6.2 de clases que contiene el paquete:

En este punto trataremos de definir el interface de la librería generada con el sistema *Siette*. La librería desarrollada ofrecerá un conjunto de clases correspondientes cada una de ellas a un tipo de pregunta distinta. A partir de estas clases, desde la pestaña de "Contenido" en la edición de una pregunta *Siette*, se crearán objetos que instancian cada clase de tipo de pregunta atendiendo a una determinada parametrización para dar lugar a una pregunta. Asimismo deberá presentar una serie de métodos de manera que puedan

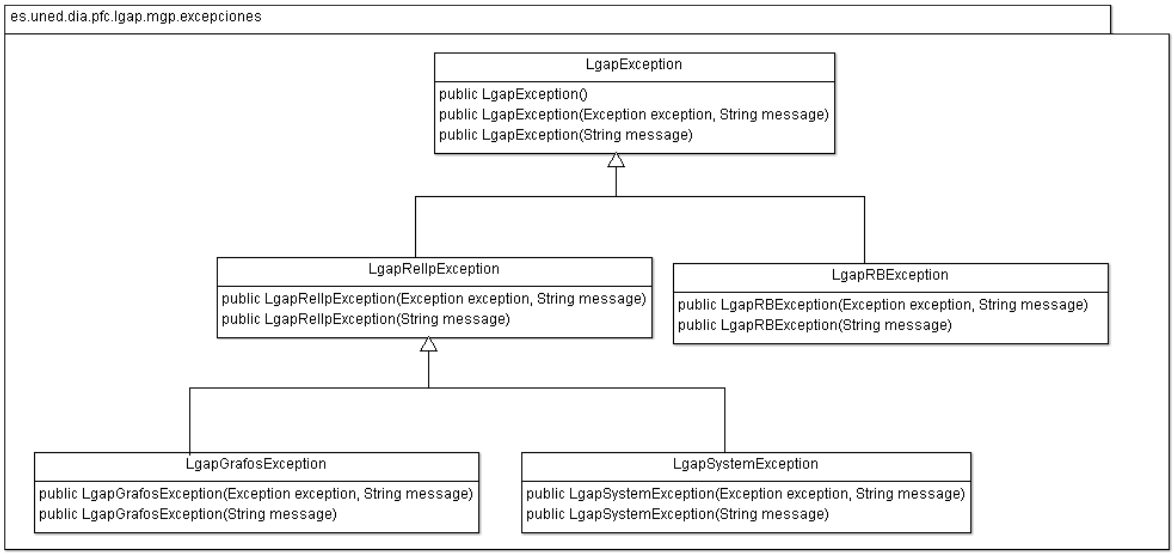


Figura 6.1: Diagrama de clases del paquete *excepciones*.

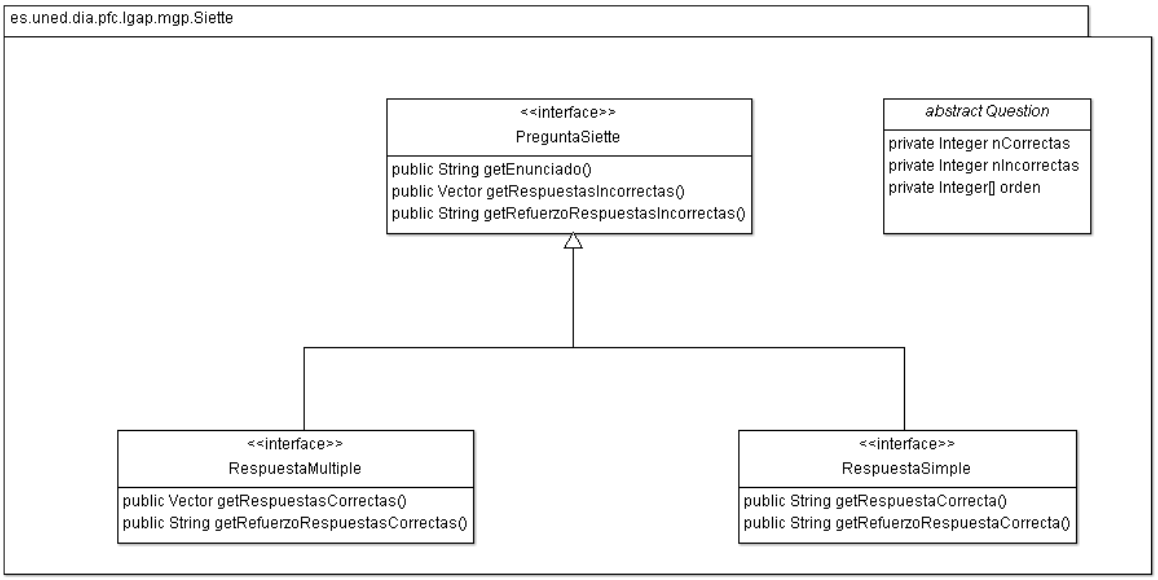


Figura 6.2: Diagrama de clases del paquete *Siette*.

ser invocados en cada apartado de la definición de cada pregunta en *Siette*. Métodos que proporcionen la respuesta correcta o respuestas correctas, respuestas incorrectas y el refuerzo asociado tanto a respuestas correctas como incorrectas. Por tanto una pregunta de test que se integre con *Siette* ha de implementar uno de los interfaces que se representan en la jerarquía dependiendo del tipo de pregunta (*RespuestaSimple*, *RespuestaMultiple*),

así como derivar de la clase abstracta *Question* que reúne los elementos comunes que se identificaron en el apartado 5.1.1.

6.1.3. Examen de Test

La librería tendrá que proporcionar la funcionalidad de generar un examen en formato PDF como una colección de preguntas de test de distintos tipos confeccionado a partir del repertorio de preguntas de los distintos tipos disponibles en la librería. Para ello se diseñará un programa cliente sencillo que tome como parámetro de entrada el número de preguntas a generar y obtenga como salida un archivo en formato PDF cuyo contenido será un examen de test compuesto por el número de preguntas distintas establecido generado a partir de los tipos de preguntas disponibles en la librería. Para ello deberá implementar el método del interfaz *ExamenTestPDF*.

En la figura 6.3 representamos el diagrama de clases correspondiente al paquete *examen*:

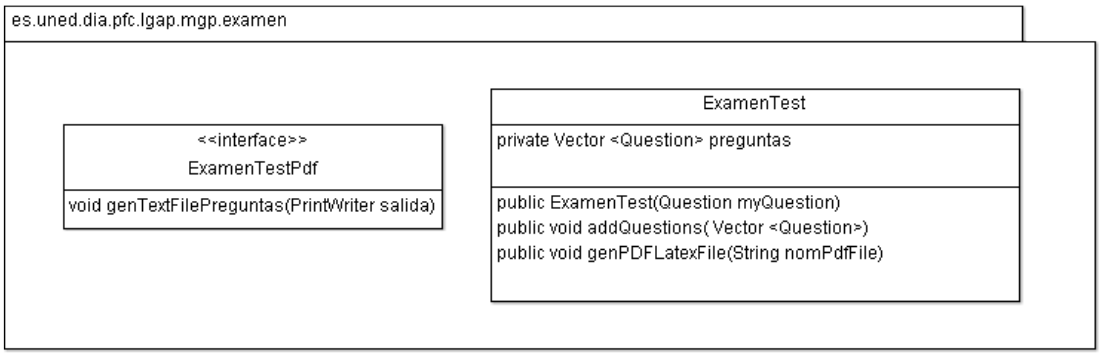


Figura 6.3: Diagrama de clases del paquete *examen*.

6.1.4. Utilidades para la generación automática

En este bloque implementaremos todas las funcionalidades que son necesarias para la generación de preguntas de forma automática. Funciones como pueden ser las siguientes:

- distintas funciones para la generación aleatoria de números reales.
- obtención el factorial de un número.

- obtener las distintas combinaciones posibles entre los valores que pueden tomar un conjunto de variables aleatorias.

- convertir un fichero en formato Latex (.tex) a PDF.

- convertir un fichero en formato SVG (.svg) a PDF.

- etc.

La figura 6.4 presenta el diagrama de clases correspondiente a este componente.

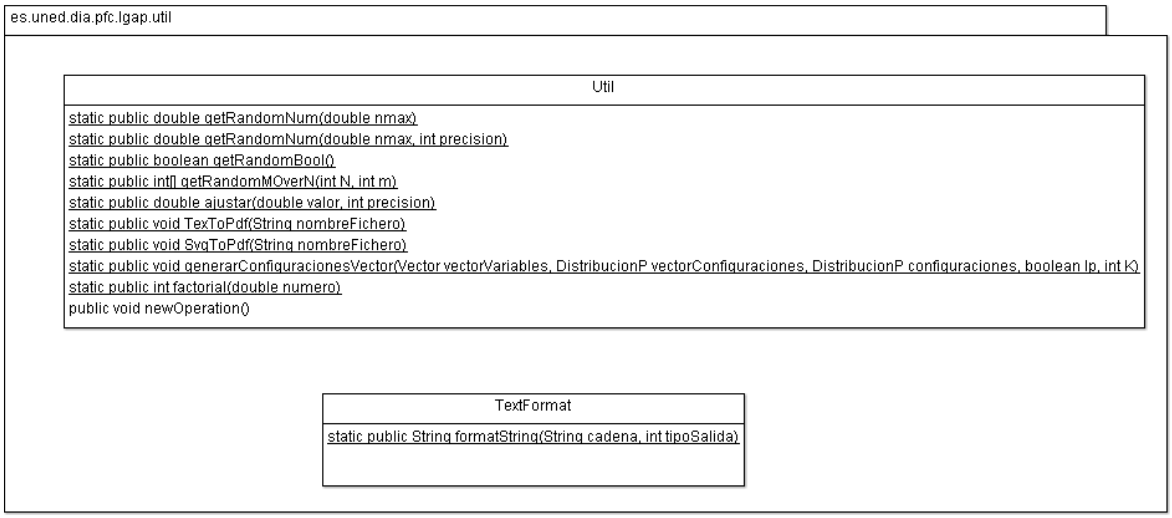


Figura 6.4: Diagrama de clases del paquete *util*.

Cada uno de los paquetes que se han definido en este apartado se pueden agrupar en un único componente o módulo software que será común al resto de componentes desarrollados en la librería para implementar cada uno de los bloques temáticos de las preguntas que se ofrecerán en el repertorio de la librería. la figura 6.5 representa dicho componente.

Los paquetes del componente "COMUNES" se completan con la definición de los paquetes *test* y *config.* El paquete *test* incluirá todos los test definidos en JUnit para las pruebas de los distintos componentes mientras que el paquete *config* reúne las utilidades que permiten leer la configuración de la librería, fichero de propiedades y logs de la aplicación.

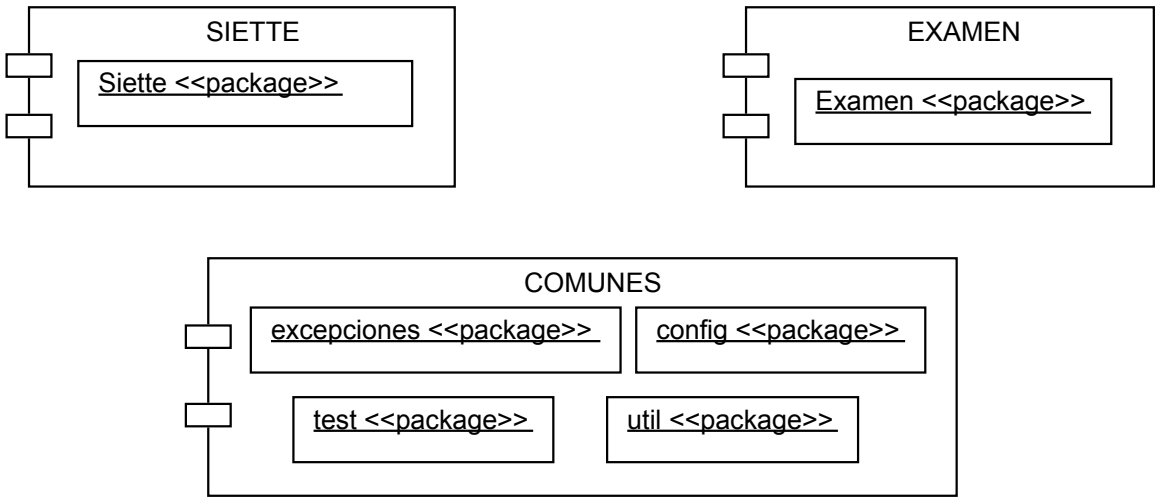


Figura 6.5: Diagrama de componentes comunes.

6.2. Preguntas tipo

En este apartado vamos a comentar cada uno de los tipos de pregunta o bloques de preguntas en los que se ha dividido la generación automática de preguntas de la librería desarrollada en el presente proyecto. Cada tipo de pregunta pertenecerá a uno de los siguientes tres bloques temáticos:

1. Relaciones de independencia probabilística o de independencia probabilística condicional entre variables aleatorias de un sistema a partir de una distribución de probabilidad.

2. Relaciones de independencia probabilística o de independencia probabilística condicional entre variables aleatorias de un sistema representadas sobre un grafo dirigido o no dirigido.

3. Inferencia en Redes Bayesianas.

En el apartado 6.1 hemos sentado las bases para poder definir los distintos tipos de preguntas de cada uno de los bloques temáticos indicados. En los apartados que siguen vamos a diseñar cada uno de los distintos tipos de preguntas que se plantean para cada bloque temático.

El usuario colaborador de nuestro sistema podrá incorporar nuevos bloques temáticos o definir nuevos tipos de pregunta para un bloque temático de los ya definidos en concreto. Pasamos a definir cada uno de los bloques por separado.

6.2.1. Bloque temático 1: Relaciones de Independencia Probabilística sobre distribución de probabilidad

En este bloque definiremos aquellas clases que tiene que ver con la definición de un sistema compuesto por un conjunto de variables aleatorias que pueden tomar un conjunto discreto de valores. Una distribución de probabilidad definida sobre las distintas combinaciones que se pueden dar entre los valores de las variables. Además las clases que nos permitan establecer las relaciones de independencia probabilística e independencia probabilística condicional entre las variables del sistema. En las figuras 6.6 y 6.7 representamos ambos diagramas de clases comentados.

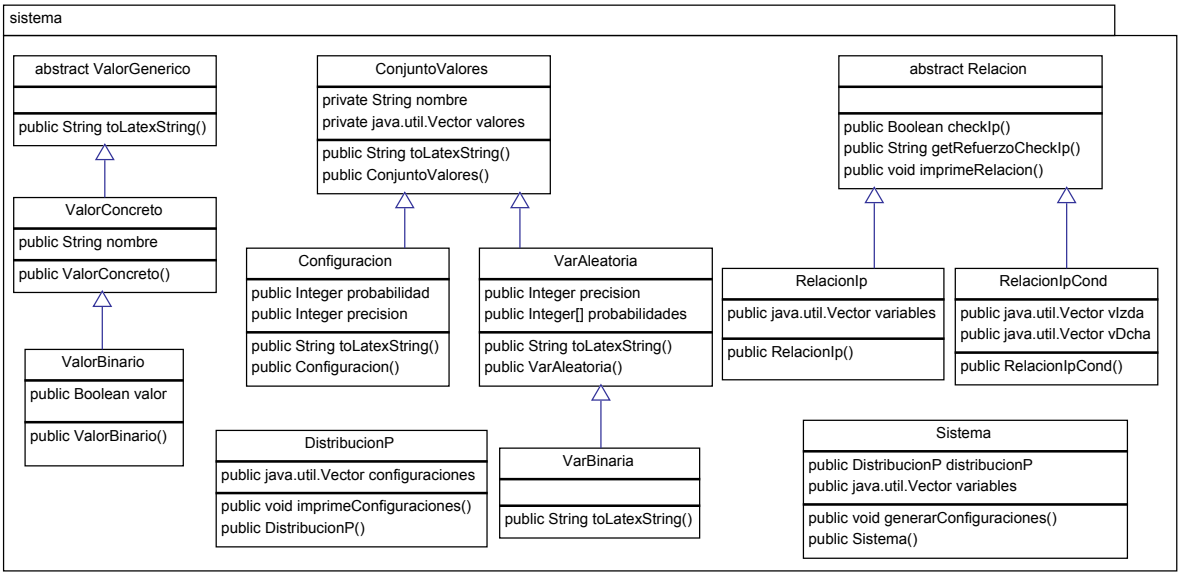


Figura 6.6: Diagrama de clases del paquete *sistema*.

Ambos diagramas son el desarrollo del componente software que se definió en el análisis realizado en el capítulo anterior representado en la figura 5.2. El paquete *sistema* define las clases que nos permiten construir el sistema sobre el que se plantean los distintos tipos de pregunta del bloque temático 1 y que veremos en detalle en las secciones siguientes de este mismo capítulo, mientras que el paquete *relipDP* reúne los cuatro tipos de preguntas que se pueden plantear.

6.2.1.1. Pregunta tipo 1

La pregunta, consistirá en determinar cuál o cuáles de las relaciones planteadas entre las variables de un sistema se dan realmente. Es decir, determinar si es verdadera o es falsa

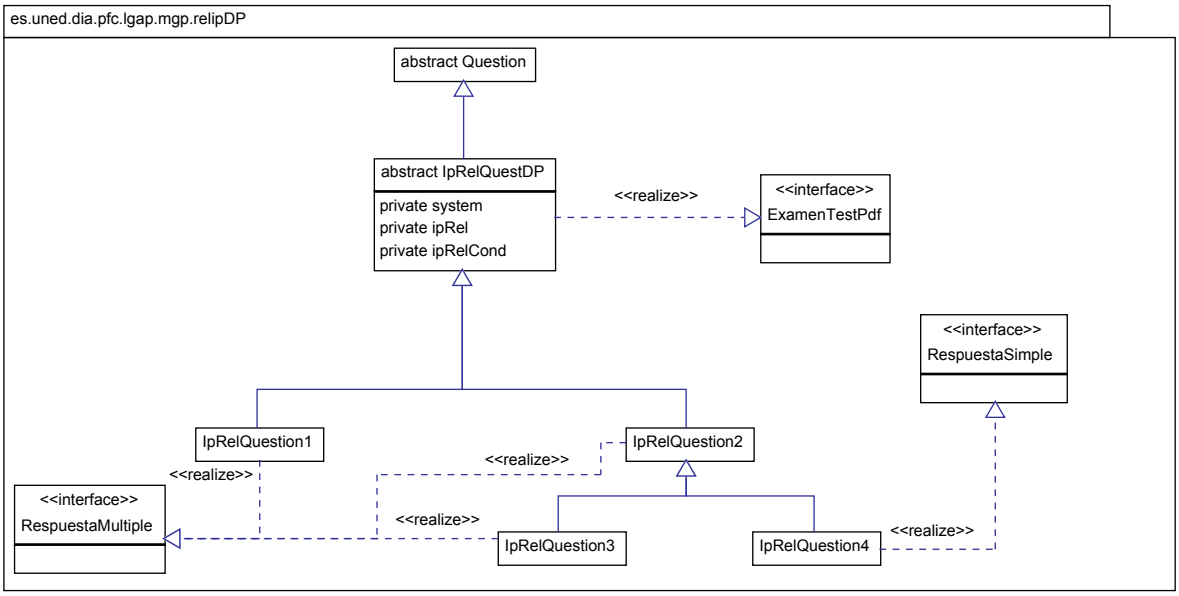


Figura 6.7: Diagrama de clases del paquete *relipDP*.

la relación especificada en cada apartado. Para la resolución de la pregunta, el alumno deberá comprobar, en base a las definiciones dadas en los apartados anteriores, si se cumple o no la relación propuesta particularizando para las variables consideradas en la pregunta. Un ejemplo, podría ser el siguiente:

Ejemplo Pregunta tipo 1 Sea una distribución de probabilidad P dada por la siguiente tabla:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,022382
$+a$	$+b$	$\neg c$	0,009522
$+a$	$\neg b$	$+c$	0,095418
$+a$	$\neg b$	$\neg c$	0,307878
$\neg a$	$+b$	$+c$	0,036518
$\neg a$	$+b$	$\neg c$	0,011178
$\neg a$	$\neg b$	$+c$	0,155682
$\neg a$	$\neg b$	$\neg c$	0,361422

Señale cuál o cuáles de las siguientes relaciones son verdaderas.

1. $I_p(A,B)$

2. $I_p\,(A, C)$
3. $I_p\,(B, C \mid A)$
4. $I_p\,(A, B \mid C)$

A partir del enunciado anterior, el alumno señalaría, en cada apartado, si la relación propuesta se cumple o no. En algunos casos se puede solicitar al sistema que de indicaciones de cómo se ha llegado a la conclusión de que una respuesta cumple la propiedad, señalando los pasos necesarios y los cálculos realizados para llegar a una conclusión final, es lo que se llama dar un **refuerzo positivo**. De la misma manera, se pueden dar indicaciones de porqué se concluye que una respuesta determinada es falsa, no cumple la propiedad especificada, señalando los motivos por los cuáles no se verifica la propiedad, es el llamado **refuerzo negativo**.

A continuación se presentan los cálculos y pasos necesarios que nos conducen a dar una respuesta correcta a cada uno de los apartados. Esta información es la que nos servirá para poder establecer cuál será el refuerzo que se le va a presentar al alumno cuando de una respuesta incorrecta o no conozca la respuesta correcta.

Solución Pregunta tipo 1 Del enunciado del ejemplo anterior, P como distribución de probabilidad, nos remite a la definición de Probabilidad Conjunta ver (4.1.1).

Particularizado para el caso del ejercicio:

- Dados $X = \{A, B, C\}$ $x =(a, b, c)$

$$\sum_x P(X) = \sum_a \sum_b \sum_c P(a, b, c) = 1$$

lo que da lugar, desarrollando la expresión, a la tabla del enunciado y se comprueba que la suma de todas las probabilidades es igual a 1.

Procedemos a comprobar si es verdadero o falso la relación del punto 1:

$I_p\,(A, B)$ nos lleva a la definición de variables independientes probabilísticamente (ver 4.1.1). Particularizado para el caso del ejercicio debemos comprobar si se cumple o no la siguiente expresión para determinar si la relación propuesta en el punto 1 es verdadera o falsa:

$\forall a, \forall b P\,(a, b) = P\,(a) \times P\,(b)$

Para comprobar si se cumple, construimos la siguiente tabla:

	$+b$	$\neg b$	$P(a)$
$+a$	$0,031904$	$0,403296$	$0,4352$
$\neg a$	$0,047696$	$0,517104$	$0,5648$
$P(b)$	$0,0796$	$0,9204$	1

A partir de los datos de la tabla del enunciado vamos rellenando la tabla anterior realizando la suma de las distintas combinaciones en las que aparece cada valor binario concreto. Por ejemplo, para obtener el valor de la combinación de valores concretos $+a+b$, sumamos los valores correspondientes de las dos primeras filas de la tabla del enunciado, prescindiendo del valor genérico c (ya que no interviene en la relación que debemos comprobar) y obtenemos el resultado $0,022382 + 0,009522 = 0,031904$. Del mismo modo vamos rellenando el resto de filas y columnas. Observar que la última fila y la última columna corresponde las probabilidades marginales (4.1.1).

Observando la tabla comprobamos, por ejemplo, que la configuración $(+a, +b)$ no cumple con la condición de independencia:

$$P(+a, +b) \neq P(+a).P(+b)$$

$$0,031904 \neq 0,4352 \times 0,0796$$

$$\boxed{\mathbf{0,031904 \neq 0,03464192}}$$

Por tanto A y B no son independientes probabilísticamente, no se cumple por tanto la relación $I_p(A, B)$ y concluimos que la relación es FALSA.

Hemos comprobado que no se cumple la independencia probabilística entre A y B para el caso $+a+b$, pero en la siguiente tabla podemos comprobar que no se cumple para ningún caso, es decir $\forall a, \forall b\, P(a, b) \neq P(a) \times P(b)$:

		$P(a,b)$	$\neq$	$P(a)$	$P(b)$	$P(a,b) \neq P(a) \times P(b)$
$+a$	$+b$	$0,031904$	$\neq$	$0,4352$	$0,0796$	$0,031904 \neq 0,03464192$
$+a$	$\neg b$	$0,403296$	$\neq$	$0,4352$	$0,9204$	$0,403296 \neq 0,40055808$
$\neg a$	$+b$	$0,047696$	$\neq$	$0,5648$	$0,0796$	$0,047696 \neq 0,04495808$
$\neg a$	$\neg b$	$0,517104$	$\neq$	$0,5648$	$0,9204$	$0,517104 \neq 0,51984192$

$I_p\,(A, C)$ Siguiendo los pasos del apartado anterior:

Particularizado para el caso del ejercicio debemos comprobar si se cumple o no la siguiente expresión para determinar si la relación propuesta en el punto 2 es verdadera o falsa:

$$\forall a, \forall c\, P\,(a, c) = P\,(a)\,.P\,(c)$$

Para comprobar si se cumple, construimos la siguiente tabla:

	$+c$	$\neg c$	$P\,(a)$
$+a$	$0,1178$	$0,3174$	$0,4352$
$\neg a$	$0,1922$	$0,3726$	$0,5648$
$P\,(c)$	$0,31$	$0,69$	1

Observando la tabla comprobamos, por ejemplo, que la configuración $(+a, +c)$ no cumple con la condición de independencia:

$$P\,(+a, +c) \neq P\,(+a)\,.P\,(+c)$$

$$0,1178 \neq 0,4352 \times 0,31$$

$$\boxed{\mathbf{0,1178 \neq 0,134912}}$$

Por tanto A y C no son independientes probabilísticamente, no se cumple por tanto la relación $I_p\,(A, C)$ y concluimos que la relación es FALSA.

Hemos comprobado que no se cumple la independencia probabilística entre A y C para el caso $+a+c$, pero en la siguiente tabla podemos comprobar que no se cumple para ningún caso, es decir $\forall a, \forall c P\,(a, c) \neq P\,(a) \times P\,(c)$:

		$P\,(a, c)$	$\neq$	$P\,(a)$	$P\,(c)$	$P\,(a, c) \neq P\,(a) \times P\,(c)$
$+a$	$+c$	$0,1178$	$\neq$	$0,4352$	$0,31$	$0,1178 \neq 0,134912$
$+a$	$\neg c$	$0,3174$	$\neq$	$0,4352$	$0,69$	$0,3174 \neq 0,300288$
$\neg a$	$+c$	$0,1922$	$\neq$	$0,5648$	$0,31$	$0,1922 \neq 0,175088$
$\neg a$	$\neg c$	$0,3726$	$\neq$	$0,5648$	$0,69$	$0,3726 \neq 0,389712$

$I_p(B, C \mid A)$ nos lleva a la definición de variables condicionalmente independientes probabilísticamente (4.1.1)

Particularizado para el caso del ejercicio:

$$\forall a, \forall b, \forall c\, P(a) > 0 \Rightarrow P(b, c \mid a) = P(b \mid a) \times P(c \mid a)$$

Comprobamos que, por ejemplo, para $(+a, +b, +c)$ no se cumple:

$$P(+b, +c \mid +a) = P(+b \mid +a) \times P(+c \mid +a)$$

$$\frac{P(+a, +b, +c)}{P(+a)} = \frac{P(+b, +a)}{P(+a)} \times \frac{P(+c, +a)}{P(+a)}$$

$$\frac{0,022382}{0,4352} = \frac{0,031904}{0,4352} \times \frac{0,1178}{0,4352}$$

$$0,0514 = 0,073309 \times 0,27068$$

$$0,0514 \neq 0,0198$$

Se concluye que las variables no son condicionalmente independientes, y por tanto no se cumple la relación $I_p(B, C \mid A)$ y por tanto concluimos que la relación es FALSA.

$I_p(A, B \mid C)$ nos lleva a la definición de variables condicionalmente independientes probabilísticamente (4.1.1).

Particularizando para el caso del ejercicio:

Lo comprobamos para:

$$P(+a, +b \mid +c) = P(+a \mid +c) \times P(+b \mid +c)$$

$$\frac{P(+a, +b, +c)}{P(+c)} = \frac{P(+a, +c)}{P(+c)} \times \frac{P(+b, +c)}{P(+c)}$$

$$\frac{0,022382}{0,31} = \frac{0,1178}{0,31} \times \frac{0,0589}{0,31}$$

$$0,0722 = 0,38 \times 0,19$$

$$\boxed{0,0722 = 0,0722} \text{ Se cumple}$$

Se cumple por tanto comprobamos si se cumple para el resto de combinaciones:

- $P\left(+a,+b \mid \neg c\right) = P\left(+a \mid \neg c\right) \times P\left(+b \mid \neg c\right)$

$$\frac{P\left(+a,+b,\neg c\right)}{P\left(\neg c\right)} = \frac{P\left(+a,\neg c\right)}{P\left(\neg c\right)} \times \frac{P\left(+b,\neg c\right)}{P\left(\neg c\right)}$$

$$\frac{0,009522}{0,69} = \frac{0,3174}{0,69} \times \frac{0,0207}{0,69}$$

$$0,0138 = 0,46 \times 0,03$$

$$\boxed{0,0138 = 0,0138} \text{ Se cumple}$$

- $P\left(+a,\neg b \mid +c\right) = P\left(+a \mid +c\right) \times P\left(\neg b \mid +c\right)$

$$\frac{P\left(+a,\neg b,+c\right)}{P\left(+c\right)} = \frac{P\left(+a,+c\right)}{P\left(+c\right)} \times .\frac{P\left(\neg b,+c\right)}{P\left(+c\right)}$$

$$\frac{0,095418}{0,31} = \frac{0,1178}{0,31} \times \frac{0,2511}{0,31}$$

$$0,3078 = 0,38 \times 0,81$$

$$\boxed{0,3078 = 0,3078} \text{ Se cumple}$$

- $P\left(+a,\neg b \mid \neg c\right) = P\left(+a \mid \neg c\right) \times P\left(\neg b \mid \neg c\right)$

$$\frac{P\left(+a,\neg b,\neg c\right)}{P\left(\neg c\right)} = \frac{P\left(+a,\neg c\right)}{P\left(\neg c\right)} \times \frac{P\left(\neg b,\neg c\right)}{P\left(\neg c\right)}$$

$$\frac{0,307878}{0,69} = \frac{0,3174}{0,69} \times \frac{0,6693}{0,69}$$

$$0,4462 = 0,46 \times 0,97$$

$$\boxed{0,4462 = 0,4462} \text{ Se cumple}$$

- $P(\neg a, +b \mid +c) = P(\neg a \mid +c) \times P(+b \mid +c)$

$$\frac{P(\neg a, +b, +c)}{P(+c)} = \frac{P(\neg a, +c)}{P(+c)} \times \frac{P(+b, +c)}{P(+c)}$$

$$\frac{0,036518}{0,31} = \frac{0,1922}{0,31} \times \frac{0,0589}{0,31}$$

$$0,1178 = 0,62 \times 0,19$$

$$\boxed{0,1178 = 0,1178} \text{ Se cumple}$$

- $P(\neg a, +b \mid \neg c) = P(\neg a \mid \neg c) \times P(+b \mid \neg c)$

$$\frac{P(\neg a, +b, \neg c)}{P(\neg c)} = \frac{P(\neg a, \neg c)}{P(\neg c)} \times \frac{P(+b, \neg c)}{P(\neg c)}$$

$$\frac{0,011178}{0,69} = \frac{0,3726}{0,69} \times \frac{0,0207}{0,69}$$

$$0,0162 = 0,54 \times 0,03$$

$$\boxed{0,0162 = 0,0162} \text{ Se cumple}$$

- $P(\neg a, \neg b \mid +c) = P(\neg a \mid +c) \times P(\neg b \mid +c)$

$$\frac{P(\neg a, \neg b, +c)}{P(+c)} = \frac{P(\neg a, +c)}{P(+c)} \times \frac{P(\neg b, +c)}{P(+c)}$$

$$\frac{0,155682}{0,31} = \frac{0,1922}{0,31} \times \frac{0,2511}{0,31}$$

$$0,5022 = 0,62 \times 0,81$$

$$\boxed{0,5022 = 0,5022} \text{ Se cumple}$$

- $P(\neg a, \neg b \mid \neg c) = P(\neg a \mid \neg c) \times P(\neg b \mid \neg c)$

$$\frac{P(\neg a, \neg b, \neg c)}{P(\neg c)} = \frac{P(\neg a, \neg c)}{P(\neg c)} \times \frac{P(\neg b, \neg c)}{P(\neg c)}$$

$$\frac{0,361422}{0,69} = \frac{0,3726}{0,69} \times \frac{0,6693}{0,69}$$

$$0,5238 = 0,54 \times 0,97$$

$$\boxed{0,5238 = 0,5238} \text{ Se cumple}$$

Por tanto se cumple $\forall a, \forall b, \forall c$ y se concluye que las variables A, B son condicionalmente dependientes respecto de C.

Por tanto la relación $I_p(A, B \mid C)$ se cumple y es VERDADERA.

6.2.1.2. Pregunta tipo 2

En el siguiente tipo de pregunta, se presentan una serie de distribuciones de probabilidad y se trata de señalar cuál o cuáles de ellas cumplen una determinada relación de independencia probabilística condicional dada. En este caso, tenemos una variación del ejercicio anterior que se resuelve de la misma manera. Un ejemplo podría ser el siguiente:

Ejemplo Pregunta tipo 2 Indique cuál de las siguientes distribuciones de probabilidad cumple la propiedad $I_p(B, C \mid A)$:

P1:

A	B	C	$P(A, B, C)$
$+a$	$+b$	$+c$	0,00459
$+a$	$+b$	$\neg c$	0,022576
$+a$	$\neg b$	$+c$	0,00561
$+a$	$\neg b$	$\neg c$	0,259624
$\neg a$	$+b$	$+c$	0,07191
$\neg a$	$+b$	$\neg c$	0,043824
$\neg a$	$\neg b$	$+c$	0,08789
$\neg a$	$\neg b$	$\neg c$	0,503976

P2:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,002574
$+a$	$+b$	$\neg c$	0,005226
$+a$	$\neg b$	$+c$	0,073508
$+a$	$\neg b$	$\neg c$	0,086292
$\neg a$	$+b$	$+c$	0,017226
$\neg a$	$+b$	$\neg c$	0,034974
$\neg a$	$\neg b$	$+c$	0,358892
$\neg a$	$\neg b$	$\neg c$	0,421308

P3:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,006902
$+a$	$+b$	$\neg c$	$0,0$
$+a$	$\neg b$	$+c$	0,033698
$+a$	$\neg b$	$\neg c$	0,2556
$\neg a$	$+b$	$+c$	0,042398
$\neg a$	$+b$	$\neg c$	$0,0$
$\neg a$	$\neg b$	$+c$	0,207002
$\neg a$	$\neg b$	$\neg c$	0,4544

P4:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,0039
$+a$	$+b$	$\neg c$	0,0117
$+a$	$\neg b$	$+c$	0,0111
$+a$	$\neg b$	$\neg c$	0,0333
$\neg a$	$+b$	$+c$	0,038352
$\neg a$	$+b$	$\neg c$	0,074448
$\neg a$	$\neg b$	$+c$	0,281248
$\neg a$	$\neg b$	$\neg c$	0,545952

Solución Pregunta tipo 2 De la misma manera que en el ejercicio anterior procederíamos a comprobar si se cumple la propiedad dada en el enunciado $I_p(B, C \mid A)$ 6.2.1.1 para todas y cada una de las distribuciones de probabilidad planteadas en el enunciado del ejemplo (6.2.1.2). Así concluiríamos que la respuesta correcta, es decir la distribución de probabilidad que satisface la propiedad dada, es la P4.

6.2.1.3. Pregunta tipo 3

En este tipo de preguntas se presentan de nuevo varias distribuciones de probabilidad y el alumno deberá indicar cuál de ellas hace que se cumplan todas las posibles relaciones tanto de independencia probabilística como de independencia probabilística condicional que se pueden dar entre las variables del sistema. Un ejemplo sería el siguiente:

Ejemplo Pregunta tipo 3 Indique para qué distribución de probabilidad de las dadas se verifican TODAS las posibles relaciones de independencia probabilística y de independencia probabilística condicional que se pueden dar entre las variables del sistema.

P1:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,022176
$+a$	$+b$	$\neg c$	0,030624
$+a$	$\neg b$	$+c$	0,024024
$+a$	$\neg b$	$\neg c$	0,033176
$\neg a$	$+b$	$+c$	0,106444
$\neg a$	$+b$	$\neg c$	0,124956
$\neg a$	$\neg b$	$+c$	0,302956
$\neg a$	$\neg b$	$\neg c$	0,355644

P2:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,014406
$+a$	$+b$	$\neg c$	0,225694
$+a$	$\neg b$	$+c$	0,014994
$+a$	$\neg b$	$\neg c$	0,234906
$\neg a$	$+b$	$+c$	0,014994
$\neg a$	$+b$	$\neg c$	0,234906
$\neg a$	$\neg b$	$+c$	0,015606
$\neg a$	$\neg b$	$\neg c$	0,244494

P3:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,00864
$+a$	$+b$	$\neg c$	0,02736
$+a$	$\neg b$	$+c$	0,01536
$+a$	$\neg b$	$\neg c$	0,04864
$\neg a$	$+b$	$+c$	0,0216
$\neg a$	$+b$	$\neg c$	0,2484
$\neg a$	$\neg b$	$+c$	0,0504
$\neg a$	$\neg b$	$\neg c$	0,5796

P4:

A	B	C	$P(A,B,C)$
$+a$	$+b$	$+c$	0,04773
$+a$	$+b$	$\neg c$	0,08127
$+a$	$\neg b$	$+c$	0,06327
$+a$	$\neg b$	$\neg c$	0,10773
$\neg a$	$+b$	$+c$	0,11137
$\neg a$	$+b$	$\neg c$	0,18963
$\neg a$	$\neg b$	$+c$	0,14763
$\neg a$	$\neg b$	$\neg c$	0,25137

Solución Pregunta tipo 3 En el sistema anterior todas las posibles relaciones de independencia probabilística y de independencia probabilística condicional serían las siguientes:

$I_p(A,B)$

$I_p(A,C)$

$I_p(B,C)$

$I_p(B,C \mid A)$

$I_p(A,C \mid B)$

$I_p(A,C \mid C)$

Para comprobar cuál o cuáles de las distribuciones de probabilidad dadas en el enunciado cumplirían todas y cada una de las relaciones posibles dadas, procederíamos a realizar la comprobación para cada distribución y propiedad de la misma manera que hemos realizado en los ejercicios anteriores. De esta manera, llegaríamos a la conclusión de que la distribución de probabilidad que satisface todas las relaciones posibles son la P2 y la P4 que daríamos como respuestas correctas a la pregunta planteada.

6.2.1.4. Pregunta tipo 4

En el siguiente tipo de preguntas se presenta una distribución de probabilidad P en el que le faltan algunos valores asociados a determinadas configuraciones de las variables del sistema (probabilidad conjunta) y se trata de averiguar qué valor le corresponde a cada fila para que se verifique una determinada relación de independencia probabilística condicional entre las variables del sistema. Un ejemplo de pregunta sería el siguiente:

Ejemplo Pregunta tipo 4 Dada la siguiente distribución de probabilidad P indique qué valor habría que poner en las filas indicadas por X_i , correspondiente a la distribución conjunta, para que se verifique la propiedad $I_p\,(B, C \mid A)$.

P:

A	B	C	$P\,(A, B, C)$
$+a$	$+b$	$+c$	0,01026
$+a$	$+b$	$\neg c$	X_1
$+a$	$\neg b$	$+c$	0,02774
$+a$	$\neg b$	$\neg c$	0,04526
$\neg a$	$+b$	$+c$	0,02592
$\neg a$	$+b$	$\neg c$	X_5
$\neg a$	$\neg b$	$+c$	0,40608
$\neg a$	$\neg b$	$\neg c$	0,43992

Solución Pregunta tipo 4 Para resolver el valor de las incógnitas planteadas en la distribución de probabilidad anterior, plantearíamos un sistema de ecuaciones a partir de las relaciones que se pueden establecer.

- Al ser una distribución de probabilidad y por la definición de probabilidad conjunta 4.1.1, tendríamos:

$X_0 + X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 = 1$

- Por otro lado, sabiendo que ha de verificar la propiedad indicada en el enunciado $I_p\,(B, C \mid A)$, y teniendo en cuenta la definición de dependencia probabilística condicional 4.1.1, podríamos plantear una serie de ecuaciones que nos permitirían calcular el valor de las incógnitas buscadas.

- Además consideramos las probabilidades marginales de cada variable. Los distintos valores que puede tomar la variable forman también una distribución de probabilidad. La suma de las probabilidades asociadas a cada valor es también la unidad. Dada la variable $X = (x_{0,...,}x_n)$, se cumple $x_0 = 1 - x_1 - ... - x_n$.

A partir de las ecuaciones que se pueden plantear en los apartados anteriores, crearíamos un sistema de ecuaciones eligiendo tantas ecuaciones como incógnitas tengamos que resolver y procederíamos a su resolución, por cualquiera de los métodos, sustitución, igualación

o reducción; para dar con la solución a la pregunta planteada. En el caso del ejemplo dado serían:

$X_1 = 0{,}01674$

$X_2 = 0{,}02808$

6.2.2. Bloque temático 2: Relaciones de Independencia Probabilística en Grafos

Los grafos son el otro sistema sobre el que vamos a plantear preguntas de independencia probabilística e independencia probabilística condicional esta vez sobre los nodos de un grafo en lugar de sobre variables aleatorias. En este tipo de preguntas nos centraremos en el temario relativo a la separación en grafos (ver apuntes de audio ApuntesAudio). Trabajaremos tanto con grafos dirigidos como no dirigidos. En las figuras 6.8 y 6.9 representamos el diagrama de clases correspondiente a cada paquete que se desarrolla en este bloque.

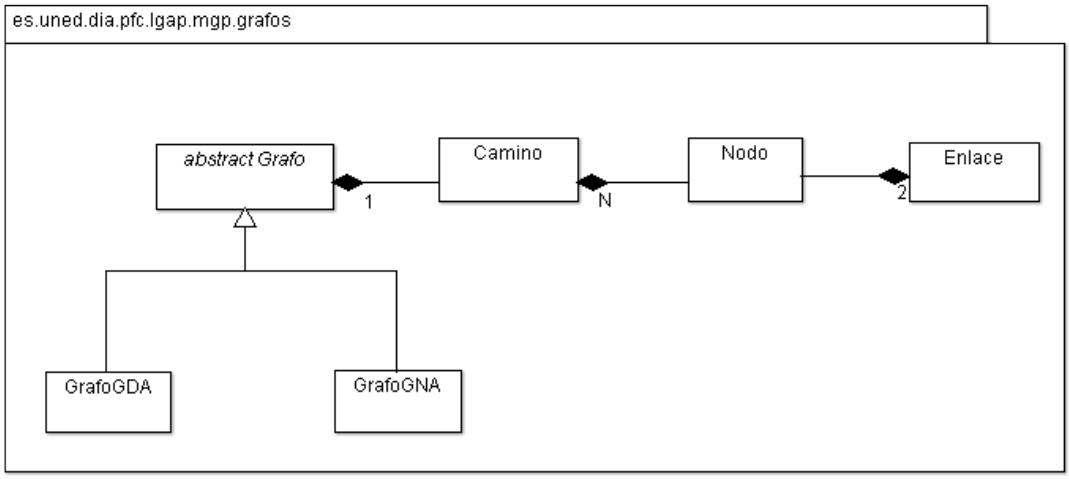


Figura 6.8: Diagrama de clases del paquete *grafos*.

Ambos diagramas son el desarrollo del componente software que se definió en el análisis realizado en el capítulo anterior representado en la figura 5.6. El paquete *grafos* define las clases que nos permiten construir el grafo sobre el que se plantean los distintos tipos de pregunta del bloque temático 2, mientras que el paquete *relipGrafo* reúne los dos tipos de preguntas que se pueden plantear y que detallamos en los siguientes apartados.

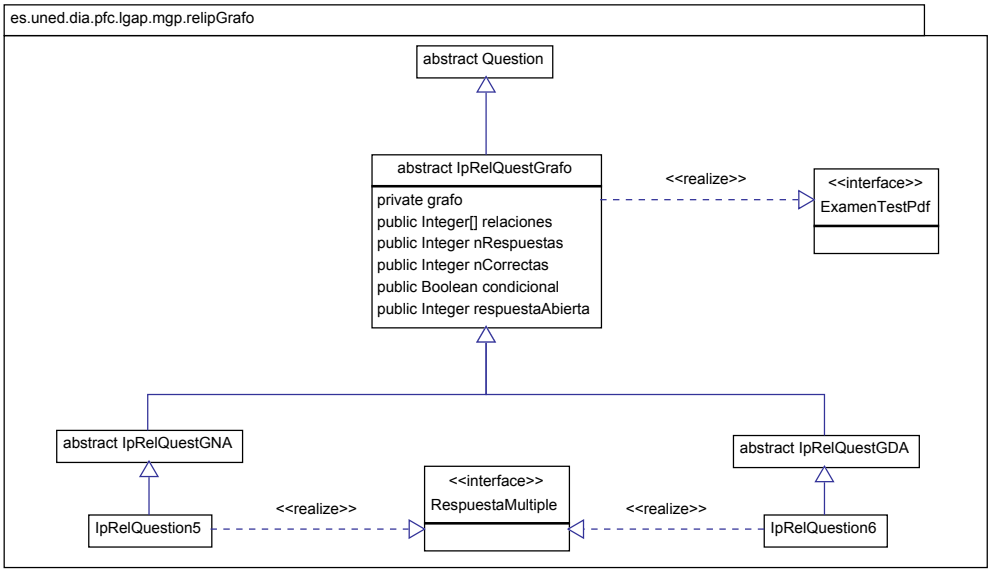


Figura 6.9: Diagrama de clases del paquete *relipGrafo*.

6.2.2.1. Pregunta tipo 5

En este tipo de preguntas partimos de la definición de un grafo no dirigido acíclico (GNA). La definición del grafo puede darse enumerando los enlaces que forman parte del mismo, o representando el grafo directamente. Se trata de indicar si es verdadera o falsa cada una de las relaciones de independencia sobre el grafo I_G que se plantean entre un par de nodos dados. Es decir se trata de ver si lo nodos indicados están o no conectados o están u-separados. Vamos a dar dos ejemplos de este tipo de preguntas:

- una que plantee relaciones entre dos nodos, relaciones de independencia probabilística.
- otra que plantee relaciones entre dos nodos dados un tercero, es decir relaciones de independencia probabilística condicional.

Pregunta tipo 5. Ejemplo 1 Sea un grafo no dirigido G que contiene cinco nodos y los siguientes enlaces: $A - B$, $A - C$, $B - C$, $B - D$ y $C - E$. Indique cuáles de las siguientes relaciones son verdaderas y cuáles son falsas, y qué caminos activos existen entre las variables de cada relación.

1. $I_G\,(A, B)$
2. $I_G\,(B, D)$

3. $I_G(A, D)$

4. $I_G(D, E)$

Solución Pregunta tipo 5. Ejemplo 1 En primer lugar construimos el grafo G a partir de los enlaces indicados en el enunciado:

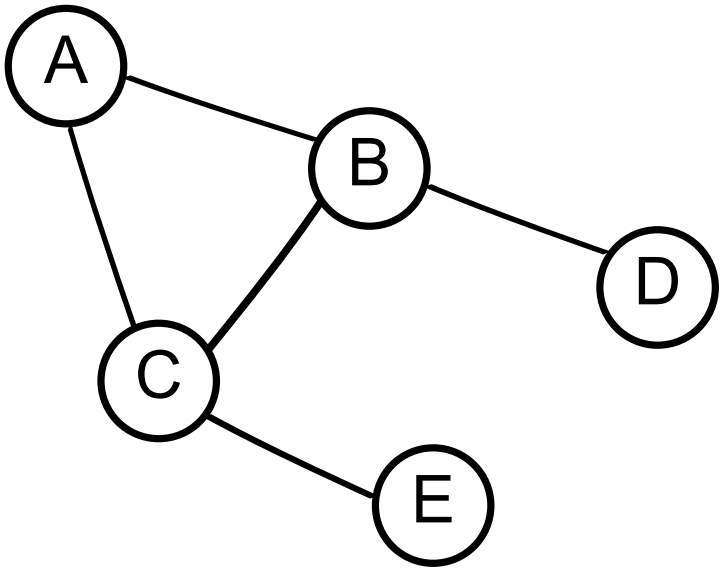


Figura 6.10: Grafo del Ejemplo 1. Pregunta tipo 5.

A continuación verificamos si se cumple cada una de las relaciones indicando cada uno de los caminos activos que existen entre las variables indicadas en la relación.

1. $I_G(A, B)$ la relación es falsa pues encontramos los siguientes caminos activos:

   a) $A - B$

   b) $A - C - B$

2. $I_G(B, D)$ la relación es falsa pues encontramos el siguiente camino activo:

   a) $B - D$

3. $I_G(A, D)$ la relación es falsa pues encontramos los siguientes caminos activos:

   a) $A - B - D$

   b) $A - C - B - D$

4. $I_G(D, E)$ la relación es falsa pues encontramos los siguientes caminos activos:

   a) $D - B - C - E$

   b) $D - B - A - C - E$

Por tanto llegamos a la conclusión de que no se cumple ninguna de las relaciones planteadas en el enunciado de la pregunta.

Pregunta tipo 5. Ejemplo 2 Sea un grafo no dirigido G que contiene cinco nodos y los siguientes enlaces: $A - B$, $A - C$, $B - C$, $B - D$ y $C - E$. Indique cuáles de las siguientes relaciones son verdaderas y cuáles son falsas, y qué caminos activos existen entre las variables de cada relación.

1. $I_G(A, B \mid C)$
2. $I_G(A, D \mid B)$
3. $I_G(D, E \mid C)$
4. $I_G(D, E \mid A)$

Solución Pregunta tipo 5. Ejemplo2 A partir de la representación gráfica del grafo dada en la figura 6.10, indicamos qué relaciones de las dadas son verdaderas o falsas señalando también para cada una de ellas qué caminos activos existen entre las dos primeras variables y si algún camino entre ellas ha sido bloqueado por la tercera variable:

1. $I_G(A, B \mid C)$ de los dos caminos activos que teníamos entre A y B, $A-B$ y $A-C-B$, el primero sigue activo y el segundo queda bloqueado por C. Por tanto la relación es falsa pues sigue existiendo un camino activo entre A y B.

2. $I_G(A, D \mid B)$ los dos caminos activos entre A y D, $A - B - D$ y $A - C - B - D$ quedan bloqueados por B por tanto se cumple la relación y podemos decir que A y D son independientes dado B.

3. $I_G(D, E \mid C)$ los dos caminos activos entre D y E, $D-B-C-E$ y $D-B-A-C-E$ quedan bloqueados por C por tanto se cumple la relación y podemos decir que D y E son independientes dado C.

4. $I_G(D, E \mid A)$de los dos caminos activos que teníamos entre D y E, $D - B - C - E$ y $D - B - A - C - E$, el primero sigue activo y el segundo queda bloqueado por A. Por tanto la relación es falsa pues sigue existiendo un camino activo entre D y E.

6.2.2.2. Pregunta tipo 6

En este tipo de preguntas partimos de la definición de un grafo dirigido acíclico (GDA). La definición del grafo puede darse enumerando los enlaces que forman parte del mismo, o representando el grafo directamente. Se trata de indicar si es verdadera o falsa cada una de las relaciones de independencia sobre el grafo I_G que se plantean entre un par de nodos dados. Es decir se trata de ver si lo nodos indicados están o no conectados o están u-separados.

Pregunta tipo 6. Ejemplo 1 Sea un grafo dirigido G que contiene seis nodos y los siguientes enlaces: $A \to C$, $B \to C$, $B \to D$, $C \to E$, $C \to F$ y $D \to F$. Indique para cada una de las siguientes relaciones si es verdadera o falsa; indique también para cada una de ellas qué caminos entre las dos variables están activos y cuáles están inactivos.

1. $I_G(A, B)$
2. $I_G(A, E)$
3. $I_G(A, D)$
4. $I_G(A, F)$
5. $I_G(D, E)$
6. $I_G(E, F)$

Solución Pregunta tipo 6. Ejemplo 1 En primer lugar construimos el grafo G a partir de los enlaces indicados en el enunciado:

A continuación para cada apartado indicamos si es verdadera o falsa la relación indicada; indicando también para cada una de ellas qué caminos entre las dos variables están activos y cuáles están inactivos.

1. $I_G(A, B)$ la relación es verdadera pues tenemos todos los caminos entre A y B inactivos.
   a) $A \to C \leftarrow B$ camino inactivo en C
   b) $A \to C \to F \leftarrow D \leftarrow B$ camino inactivo en F
2. $I_G(A, E)$ la relación es falsa pues tenemos un camino activo.
   a) $A \to C \to E$ camino activo

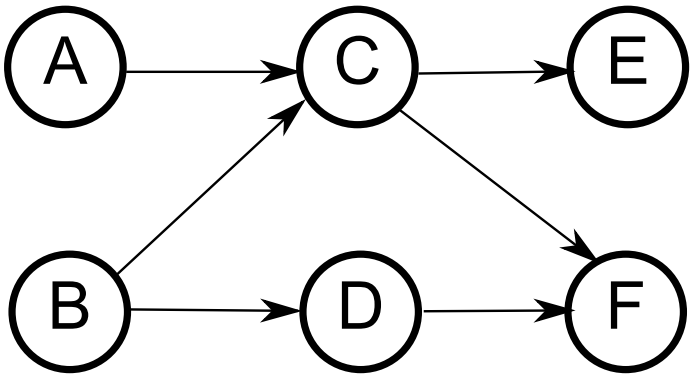


Figura 6.11: Grafo del Ejemplo 1. Pregunta tipo 6.

b) $A \rightarrow C \rightarrow F \leftarrow D \leftarrow B \rightarrow C \rightarrow E$ camino inactivo bloqueado por F

3. $I_G(A, D)$ la relación es verdadera pues todos los caminos entre las variables A y D están inactivos:

   a) $A \rightarrow C \rightarrow F \leftarrow D$ camino inactivo en F

   b) $A \rightarrow C \leftarrow B \rightarrow D$ camino inactivo en C

4. $I_G(A, F)$ la relación es falsa pues tenemos un camino activo entre las variables A y F :

   a) $A \rightarrow C \rightarrow F$ camino activo

   b) $A \rightarrow C \leftarrow B \rightarrow D \rightarrow F$ camino inactivo bloqueado en C

5. $I_G(D, E)$ la relación es falsa pues tenemos un camino activo entre las variables D y E :

   a) $D \rightarrow F \leftarrow C \rightarrow E$ camino inactivo bloqueado en F

   b) $D \leftarrow B \rightarrow C \rightarrow E$ camino activo

6. $I_G(E, F)$ la relación es falsa pues tenemos dos caminos activos.

a) $E \leftarrow C \rightarrow F$ camino activo

b) $E \leftarrow C \leftarrow B \rightarrow D \rightarrow F$ camino activo

Pregunta tipo 6. Ejemplo 2 Indique cuáles de las siguientes relaciones son verdaderas y cuáles son falsas. Indique también para cada una de ellas si algún camino entre las dos primeras variables que estaba inactivo ha sido activado por la tercera, y viceversa, es decir, si algún camino que estaba activo ha sido bloqueado.

1. $I_G(A, B \mid C)$
2. $I_G(A, B \mid E)$
3. $I_G(A, B \mid F)$
4. $I_G(A, D \mid C)$
5. $I_G(A, D \mid F)$
6. $I_G(A, F \mid C)$
7. $I_G(A, F \mid E)$
8. $I_G(D, E \mid B)$
9. $I_G(E, F \mid C)$

Solución Pregunta tipo 6. Ejemplo 2. A partir de la representación gráfica del grafo dada en la figura6.2.2.2, indicamos cuáles de las siguientes relaciones son verdaderas y cuáles son falsas, indicando también para cada una de ellas si algún camino entre las dos primeras variables que estaba inactivo ha sido activado por la tercera, y viceversa, es decir, si algún camino que estaba activo ha sido bloqueado.

1. $I_G(A, B \mid C)$ la relación es falsa pues

   a) $A \rightarrow C \leftarrow B$ el camino que estaba inactivo ha sido activado por C

   b) $A \rightarrow C \rightarrow F \leftarrow D \leftarrow B$ el camino que estaba activo entre A, C y F ha sido bloqueado por la tercera variable C

2. $I_G(A, B \mid E)$ es verdadera pues:

   a) $A \rightarrow C \leftarrow B$ el camino que estaba inactivo sigue inactivo.

b) $A \rightarrow C \rightarrow F \leftarrow D \leftarrow B$ el camino que estaba inactivo en F sigue inactivo

3. $I_G(A, B \mid F)$ no se cumple pues:

   a) $A \rightarrow C \leftarrow B$ el camino que estaba inactivo sigue inactivo.

   b) $A \rightarrow C \rightarrow F \leftarrow D \leftarrow B$ el camino que estaba inactivo pasa a estar activado por la tercera variable F

4. $I_G(A, D \mid C)$ falsa

   a) $A \rightarrow C \rightarrow F \leftarrow D$ el camino que estaba inactivo en F queda ahora también bloqueado por la tercera variable C

   b) $A \rightarrow C \leftarrow B \rightarrow D$ el camino que estaba inactivo en C ha sido activado por la tercera variable.

5. $I_G(A, D \mid F)$ falsa pues

   a) $A \rightarrow C \rightarrow F \leftarrow D$ el camino que estaba inactivo ha sido activado por la tercera variable F

   b) $A \rightarrow C \leftarrow B \rightarrow D$ camino inactivo que sigue inactivo

6. $I_G(A, F \mid C)$ no se cumple pues

   a) $A \rightarrow C \rightarrow F$ el camino que estaba activo ha sido bloqueado por la tercera variable C

   b) $A \rightarrow C \leftarrow B \rightarrow D \rightarrow F$ el camino que estaba inactivo ha sido activado por la tercera variable C

7. $I_G(A, F \mid E)$ no se cumple pues

   a) $A \rightarrow C \rightarrow F$ camino activo que sigue activo

   b) $A \rightarrow C \leftarrow B \rightarrow D \rightarrow F$ camino inactivo en C que sigue inactivo

8. $I_G(D, E \mid B)$ verdadera pues

   a) $D \rightarrow F \leftarrow C \rightarrow E$ camino inactivo que sigue inactivo

   b) $D \leftarrow B \rightarrow C \rightarrow E$ el camino que estaba activo ha sido bloqueado por la tercera variable B

9. $I_G(E, F \mid C)$ verdadera pues

   a) $E \leftarrow C \rightarrow F$ el camino que estaba activo ha sido bloqueado por la tercera variable C

   b) $E \leftarrow C \leftarrow B \rightarrow D \rightarrow F$ el camino que estaba activo ha sido bloqueado por la tercera variable C

6.2.3. Bloque temático 3: Inferencia en Redes Bayesianas

En este tercer bloque temático vamos a plantear preguntas sobre inferencia en redes bayesianas. Para ello vamos a hacer uso de la librería *OpenMarkov* comentada con anterioridad. Partiremos de la estructura gráfica de una red bayesiana que constituye su parte cualitativa, junto con su parte cuantitativa y la descripción de las variables aleatorias representadas en cada nodo del grafo. A partir de aquí se introducirá evidencia en la red mediante la incorporación de uno o varios hallazgos en los valores que pueden tomar cada una de la variables aleatorias para pasar a determinar las probabilidades a posteriori de alguna o algunas variables de interés dentro de la red.

La figura 6.12 representa el paquete *rb* que junto con el API ofrecido por la librería *OpenMarkov* nos va a permitir definir los tres tipos distintos de tipos de pregunta que se presentan en este tercer bloque temático.

En los apartados siguientes presentamos cada una de las variantes o tipo de preguntas relacionadas con inferencia en redes bayesianas.

6.2.3.1. Pregunta tipo 7

En este tipo de preguntas trabajamos con la red bayesiana *Enfermedad-Test*. Se dan como datos de la pregunta la prevalencia, la sensibilidad y la especificidad. El alumno ha de calcular los valores predictivos del test para dicha enfermedad. A continuación mostramos un ejemplo de pregunta de este tipo.

Pregunta tipo 7. Ejemplo. Para una enfermedad X cuya prevalencia es del 0,08 % existe un test Y con una sensibilidad del 0,75 % y una especificidad del 0,96 % ¿Cuáles son los valores predictivos del test para dicha enfermedad?

Si se sabe que el resultado del test ha dado positivo. ¿Cuál es la probabilidad a posteriori de padecer la enfermedad?

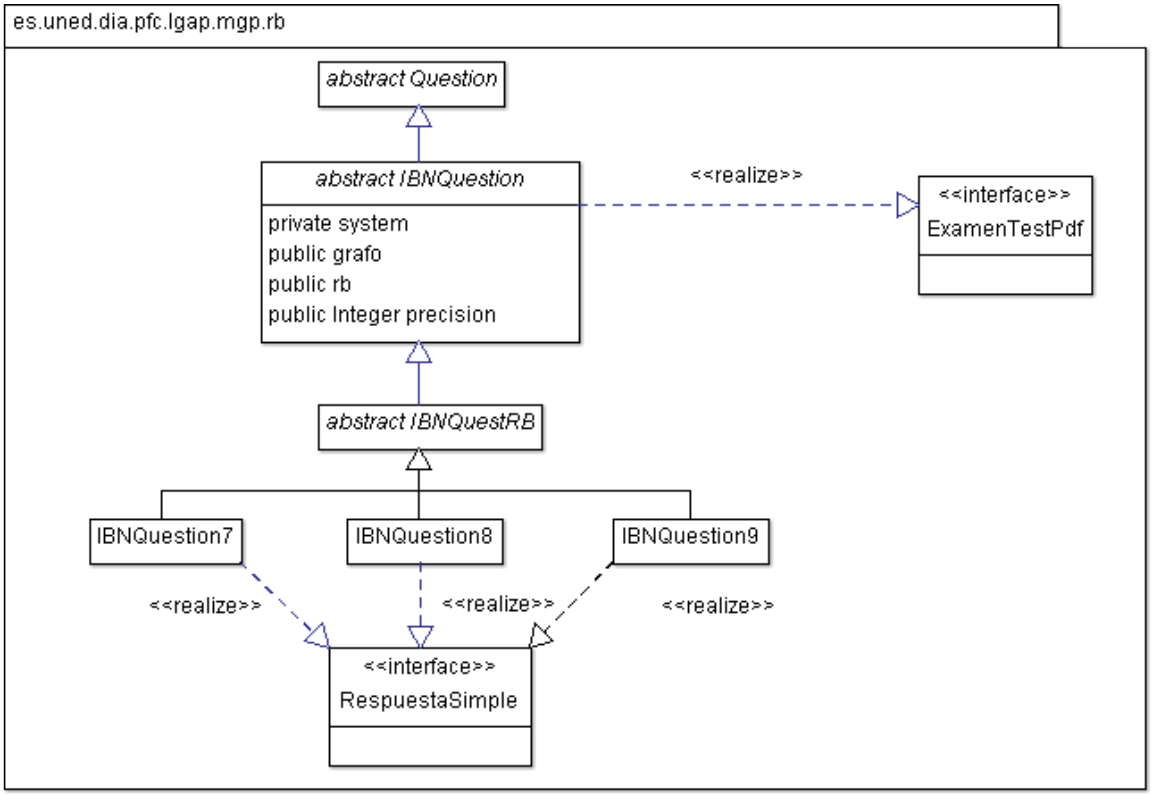


Figura 6.12: Diagrama de clases del paquete *rb*.

Solución Pregunta tipo 7. Ejemplo. A partir de los datos del enunciado generamos la tabla de probabilidad condicionada siguiente:

X	Y	$P(A, B, C)$
$+x$	$+y$	$0,06$
$+x$	$\neg y$	$0,02$
$\neg x$	$+y$	$0,0368$
$\neg x$	$\neg y$	$0,8832$

El valor de la primera fila se obtiene del producto de la sensibilidad por la prevalencia.

$$Sensibilidad = P(+y|+x)$$

$$P(+y|+x) = \frac{P(+y,+x)}{P(+x)}$$

$$P(+y,+x) = P(+y|+x) \times P(+x)$$

$$P(+x, +y) = 0,75 \times 0,08$$

$$P(+x, +y) = 0,06$$

El valor de la segunda fila se obtiene como el producto de la prevalencia por el complementario de la sensibilidad o bien teniendo en cuenta que la suma de las dos primeras filas ha de ser la prevalencia $P(+x)$. Por tanto:

$$P(+x, \neg y) = P(+x) - P(+x, +y)$$

$$P(+x, \neg y) = 0,08 - 0,06$$

$$P(+x, \neg y) = 0,02$$

El valor de la cuarta fila se obtiene del producto de la especificidad por el complementario de la prevalencia.$P(\neg x) = 1 - P(+x)$. Por tanto:

$$Especificidad = P(\neg y|\neg x)$$

$$P(\neg y|\neg x) = \frac{P(\neg y, \neg x)}{P(\neg x)}$$

$$P(\neg y, \neg x) = P(\neg y|\neg x) \times P(\neg x)$$

$$P(\neg x, \neg y) = 0,92 \times 0,96$$

$$P(+x, +y) = 0,8832$$

El valor de la tercera fila se obtiene del producto del complementario de la especificidad por el complementario de la prevalencia o bien teniendo en cuenta que la suma de las dos últimas filas ha de ser el complementario de la prevalencia.

$$P(\neg x, +y) = P(\neg x) - P(\neg x, \neg y)$$

$$P(\neg x, +y) = 0,92 - 0,8832$$

$$P(+x, \neg y) = 0,0368$$

El gafo que representa la Red Bayesiana:

Figura 6.13: Red Bayesiana *Disease-Test*.

Una vez completada la tabla de probabilidad condicionada podemos obtener las probabilidades buscadas que representan los valores predictivos del test:

$$Y(+y, \neg y) = (0{,}0968, 0{,}9032)$$

El valor correspondiente a $P(+y)$ se obtiene sumando la primera y tercera fila. El valor correspondiente a $P(\neg y)$ se obtiene sumando la segunda y cuarta fila.

Para contestar a la segunda cuestión, introducimos el hallazgo $Y = +y$

Evidencia: Y: positive

Probabilidad a posteriori: X:0,61983

que obtenemos resolviendo la expresión del Teorema de Bayes para la obtención del Valor Predictivo Positivo (VPP) siguiente:

$$VPP = \frac{Sens \times Prev}{Sens \times Prev + (1 - Espec) \times (1 - Prev)}$$

6.2.3.2. Pregunta tipo 8

En este tipo de preguntas la red bayesiana que se plantea en el enunciado será una de las ofrecidas en la factoría incorporada en el programa *OpenMarkov*:

- red ***bN_ABC*** será una red formada por tres nodos (A, B, C) y tres enlaces $(A \to B, A \to C, B \to C)$. Se presenta en el enunciado el grafo que representa la red junto con los posibles estados que puede tomar cada una de las variables que la conforman. Además se da también la parte cuantitativa, es decir las tablas de probabilidades condicionadas que definen la red. A partir ahí se introduce evidencia en alguno de los nodos de la red y se pregunta al alumno por la probabilidad asociada a alguno de los valores de una variable de interés de la red elegida al azar. Se presentan las distintas alternativas entre las cuáles sólo una de ellas representa la probabilidad buscada.

- red ***bN_XYZ***, al igual que en el caso anterior se parte de la estructura de la red ahora formada por tres nodos (X, Y, Z) y dos enlaces $X \to Y, X \to Z$. Se describe la red a partir de los distintos estados de las variables y de la parte cuantitativa de la misma y se pregunta por la probabilidad asociada a algún valor de la variable de interés (elegida de manera aleatoria) una vez que se ha introducido evidencia en la red mediante la inclusión de uno o dos hallazgos en alguno de los nodos que la conforman.

- la red ***Asia*** es una red típica de estudio en la literatura de los modelos gráficos probabilistas. La Figura 6.14 muestra el grafo acíclico dirigido correspondiente a la red causal Asia que modela el siguiente problema: La tuberculosis (T) y el cáncer de pulmón (L) son causas de que un paciente esté enfermo del pulmón $(TOrC)$. Si una persona está enferma del pulmón, entonces la enfermedad puede provocarle disnea (D) y también pueden influir en el resultado de una prueba de rayos X en el pecho (X). Por otra parte, la bronquitis (B) es otra causa de disnea. Además, el hecho de haber visitado recientemente Asia (A) incrementa la probabilidad de padecer tuberculosis, mientras que el hecho de ser fumador (S) es una de las causas posibles de la bronquitis y del cáncer de pulmón.

Al igual que en los dos casos anteriores, el alumno deberá elegir entre las opciones propuestas, cuál de ellas se corresponde a la probabilidad a posteriori de que una determinada variable o nodo de la red tome ese valor una vez que se ha introducido cierta evidencia en la red.

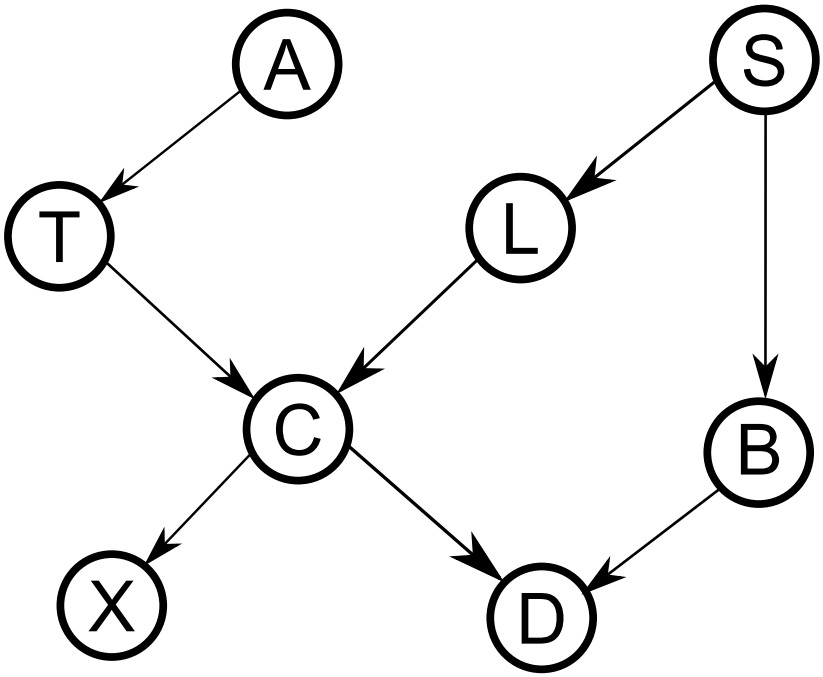


Figura 6.14: Red Bayesiana *Asia*.

Pregunta tipo 8. Ejemplo. Sea la red bayesiana de la figura 6.15:

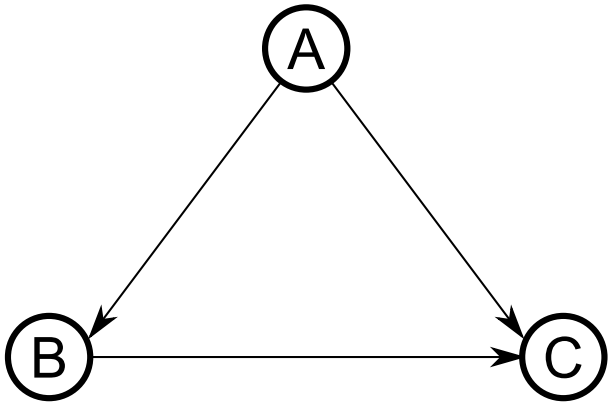


Figura 6.15: Red Bayesiana *bN_ABC*.

Con los siguientes estados para cada una de las variables o nodos de la red:

$A = (present; absent)$

$B = (present; absent)$

$C = (present; absent)$

Las tablas de probabilidades condicionadas que definen la red son las siguientes:

$P(A) = \{0,8; 0,2\}$

$P(B \mid A) = \{0,1; 0,9; 0,3; 0,7\}$

$P(C \mid A, B) = \{0,02; 0,98; 0,71; 0,29; 0,16; 0,84; 0,85; 0,15\}$

Si se introduce en la variable B el hallazgo "absent".

¿Cuál es la probabilidad asociada al valor "absent" de la variable A?

1. 0,3843
2. 0,6308
3. 0,1628
4. 0,1983

Solución Pregunta tipo 8. Ejemplo. A partir de las tablas de probabilidades condicionadas dadas en el enunciado, obtenemos la siguiente distribución de probabilidad:

A	B	C	$P(A, B, C)$
$+a$	$+b$	$+c$	$0,0016$
$+a$	$+b$	$\neg c$	$0,0784$
$+a$	$\neg b$	$+c$	$0,1152$
$+a$	$\neg b$	$\neg c$	$0,6048$
$\neg a$	$+b$	$+c$	$0,0426$
$\neg a$	$+b$	$\neg c$	$0,0174$
$\neg a$	$\neg b$	$+c$	$0,119$
$\neg a$	$\neg b$	$\neg c$	$0,21$

Las probabilidades a posteriori de cada variable, calculadas a partir de la tabla dada son:

$$A \rightarrow P(A) = \{0{,}8, 0{,}2\}$$
$$B \rightarrow P(B) = \{0{,}14, 0{,}86\}$$
$$C \rightarrow P(C) = \{0{,}278, 0{,}722\}$$

que se obtienen de la proyección de cada valor de cada variable.

La probabilidad asociada al valor "absent" de la variable A vendrá dada por el cociente:

$$P(\neg a) = \frac{proyeccion(\neg b, \neg a)}{proyeccion(\neg b)}$$

Ahora si tomamos la proyección de $(\neg b, \neg a)$ sobre la distribución obtenemos el valor 0,14

Si tomamos la proyección de $\neg b$ en la distribución de probabilidad dada, obtenemos el valor 0,86.

El resultado final se obtiene como el cociente de ambas $0,14/0,86 = 0,1628$

Por lo tanto la opción correcta es la 4.

6.2.3.3. Pregunta tipo 9

En este último tipo de pregunta la red para la que se plantea la pregunta es una red de entre 2 y 8 nodos que se genera de manera aleatoria. El planteamiento es el mismo que en los apartados anteriores, es decir se trata de calcular una determinada probabilidad tras la evidencia introducida en la red. A continuación mostramos un ejemplo de pregunta de este tipo resultado de una ejecución de la librería.

Pregunta tipo 9. Ejemplo. Sea la red bayesiana de la figura 6.16:

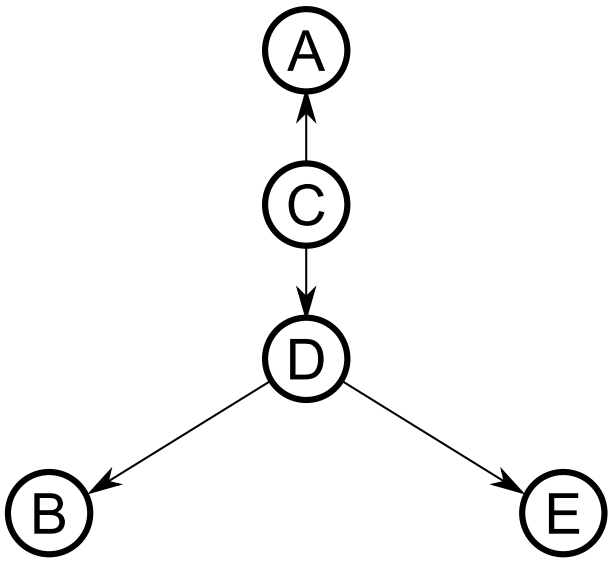


Figura 6.16: Red Bayesiana generada aleatoriamente.

Con los siguientes estados para cada una de las variables o nodos de la red:

$$A = (yes; no)$$
$$B = (present; absent)$$
$$C = (yes; no)$$
$$D = (yes; no)$$
$$E = (yes; no)$$

Las tablas de probabilidades condicionadas que definen la red son las siguientes:

$$P(A|C) = \{0,182; 0,818; 0,086; 0,914\}$$
$$P(B|D) = \{0,097; 0,903; 0,017; 0,983\}$$
$$P(C) = \{0,224; 0,776\}$$
$$P(D|C) = \{0,237; 0,763; 0,348; 0,652\}$$
$$P(E|D) = \{0,329; 0,671; 0,222; 0,778\}$$

Si se introduce en las variables B, C los hallazgos "*present*", "*no*" respectivamente. ¿Cuál es la probabilidad asociada al valor "yes" de la variable A?

1. 0,409
2. 0,376
3. 0,086
4. 0,46

Solución Pregunta tipo 9. Ejemplo. Para llegar a la solución a la pregunta seguimos el mismo procedimiento explicado en el apartado anterior respecto a las preguntas de tipo 86.2.3.2A partir de la distribución de probabilidad conjunta correspondiente a la factorización de la red, obtenemos la probabilidad buscada como el cociente siguiente:

- la probabilidad conjunta (proyecciones) de los valores de los hallazgos introducidos en la red junto con el valor buscado en el numerador
- los valores de los hallazgos introducidos en la red (la evidencia introducida) en el denominador.

Para el caso del ejemplo, tendríamos la siguiente expresión:

$$P(+a) = \frac{proyeccion(+b, \neg c, +a)}{proyeccion(+b, \neg c)}$$

6.3. Diagrama de componentes completo

En este apartado, a modo de resumen final del capítulo, vamos a presentar el diagrama de componentes completo que se ha ido diseñando a lo largo de esta etapa del ciclo de vida del proyecto. La figura 6.17 muestra el diagrama completo.

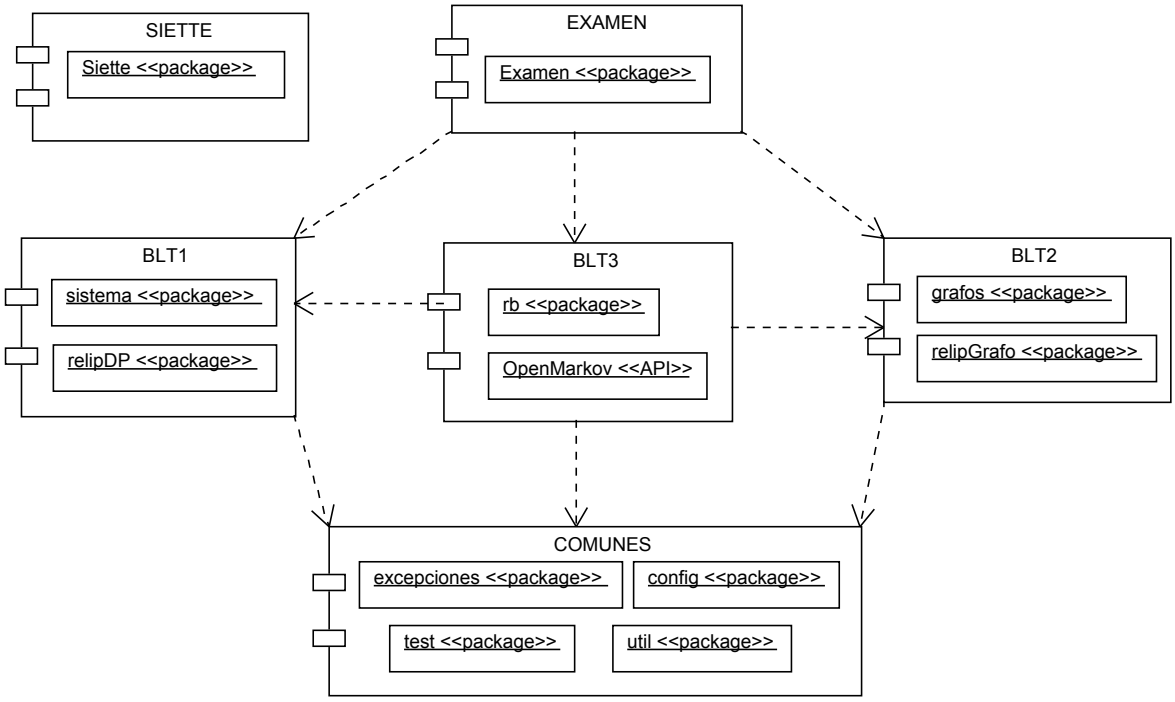


Figura 6.17: Diagrama de componentes completo.

Capítulo 7

Construcción e implementación

En este capítulo se explicará detalladamente el desarrollo realizado para la implementación de la **L**ibrería Java de **G**eneración **A**utomática de **P**reguntas *LGAP* sobre **M**odelos **G**ráficos **P**robabilistas *MGP*. Empezaremos con la parte común que constituye el *framework* base de desarrollo de los distintos tipos de preguntas. Para ello se parte, en primer lugar, de los diagramas de clases que constituyen el diseño realizado en el capítulo anterior para plantear el problema a resolver y a continuación se comentan brevemente cada una de las clases implementadas y sus métodos más representativos. Los diagramas de clases, constituyen la parte estática del diseño, lo incluimos en este capítulo de implementación como introducción para presentar el desarrollo realizado en cada uno de los paquetes en que se se ha dividido la librería *LGAP*.

Para cada bloque temático en el que se enmarca cada tipo de pregunta, se ha realizado un diseño *"bottom-up"*, es decir, se ha partido de las clases más básicas, que constituyen los elementos fundamentales o componentes elementales, hasta llegar a la construcción de un sistema completo, a partir del cuál, se plantearán los distintos tipos de preguntas relacionadas con modelos gráficos probabilistas. A continuación se especifica el sistema o elementos que se consideran para cada bloque temático:

- **bloque temático 1** que plantea relaciones de independencia probabilística sobre una distribución de probabilidad. En este caso la definición del sistema sobre el que se plantean las preguntas consiste en un conjunto de variables aleatorias y una serie de relaciones de independencia probabilística que se pueden establecer entre ellas a partir de una distribución de probabilidad definida sobre las distintas configuraciones, es decir sobre las distintas combinaciones que se pueden dar entre los valores de cada variable del sistema. Esta parte será la base para establecer la parte cuantitativa de una red bayesiana y que nos permitirá definir las preguntas

en el bloque temático 3 sobre inferencia en redes bayesianas.

- el **bloque temático 2** trata las relaciones de independencia probabilística sobre grafos (dirigidos o no dirigidos). En este caso el sistema es un grafo formado por un conjunto de elementos más básicos como son los nodos y los enlaces. Son precisamente los enlaces los que representan las relaciones entre los nodos o variables. Este bloque temático servirá para definir la parte cualitativa de una red bayesiana y que será útil para el tercer bloque temático que comentamos a continuación.

- el **tercer** y último bloque temático que vamos a considerar será la inferencia en redes bayesianas y el sistema sobre el que se plantean las preguntas es la propia red bayesiana formada por un grafo (parte cualitativa de la red) sobre el que se ha definido una distribución de probabilidad (parte cuantitativa). Aquí recurrimos a la definición dada en el capítulo de Inferencia en Redes Bayesianas4.3. Para este último bloque utilizaremos como *API* el programa para construcción y evaluación de modelos gráficos probabilistas *OpenMarkov* (OpenMarkov), en particular de la parte que nos permite construir redes bayesianas y realizar inferencias sobre la red a partir de la evidencia introducida.

En los apartados siguientes vamos a comentar cada uno de los paquetes en los que han sido agrupadas las clases en el desarrollo de la librería de generación automática de preguntas sobre modelos gráficos probabilistas *LGAPMGP*.

7.1. Paquetes comunes

En este primer apartado vamos a comentar los paquetes que forman el *framework* básico de desarrollo para el resto de paquetes de la librería. En este paquete definiremos los elementos comunes al resto de paquetes de la librería.

7.1.1. Paquete *excepciones*

En la librería implementada se ha definido una jerarquía de clases propia para el control de excepciones de la aplicación dentro del paquete nombrado como *es.uned.dia.pfc.lgap.mgp.excepciones* (ver 6.1.1). Esto nos permitirá mostrar al usuario de la librería una información más completa sobre los posibles errores que se puedan producir durante la utilización de la misma. Comenzaremos explicando cada una de las clases que componen el paquete por separado.

7.1.1.1. Clase *LgapException*

Constituirá la clase base de todas las excepciones que se definen para la librería de generación automática de preguntas. Será por tanto la clase más general de todas y la que hereda directamente de la clase Java *Exception*.

7.1.1.2. Clase *LgapRelIpException*

Clase definida para todas aquellas excepciones que tienen que ver con relaciones de independencia probabilística y de independencia probabilística condicional. La clase especializa la excepción heredando de la clase *LgapException* comentada en el apartado anterior. Este tipo de excepciones se producirán cuando se dé alguna circunstancia anómala en la construcción de una relación de independencia probabilística entre dos variables de un sistema o de una relación de independencia probabilística condicional entre dos variables aleatorias respecto de una tercera. El sistema puede ser un conjunto de variables aleatorias o bien un grafo acíclico tanto dirigido como no dirigido. Esto dará lugar a los dos tipos de excepciones que se comentarán en los apartados siguientes y que especializan esta clase.

7.1.1.3. Clase *LgapSystemException*

En esta clase se englobarían todas aquellas excepciones que se producen en el sistema compuesto por un conjunto de variables aleatorias en el que se ha definido una distribución de probabilidad entre los distintos valores que pueden adoptar las variables, es decir entre las distintas configuraciones del sistema.

7.1.1.4. Clase *LgapGrafosException*

En esta clase se englobarían todas aquellas excepciones que se producen en el sistema compuesto por un conjunto de variables aleatorias definidas como los nodos de un grafo acíclico dirigido o no dirigido. Los enlaces entre los nodos representan las relaciones que se establecen entre las variables.

7.1.1.5. Clase *LgapRBException*

En el caso de esta clase, se agruparían las excepciones propias en la inferencia en redes bayesianas.

7.1.2. Paquete *util*

En el apartado 6.1.4 presentamos el diagrama de clases del paquete *es.uned.dia.pfc.lgap.mgp.util*. Este paquete contendrá todas las funcionalidades comunes al resto de paquetes y que nos van a servir para realizar tareas comunes en el planteamiento de las preguntas que tienen que ver con métodos gráficos probabilísticos.

En este paquete definimos la clase *Util* que contendrá todas las funcionalidades comunes al resto de las clases del sistema. Además la clase *TextFormat* nos servirá para aplicar al texto de las preguntas el formato adecuado en función del tipo de salida. En los apartados siguientes comentamos cada una de estas clases por separado.

7.1.2.1. Clase Util

Clase de utilidades. En esta clase vamos a disponer de una serie de métodos comunes que nos serán útiles para los cálculos y operaciones que se repiten en las distintas clases que forman parte de nuestra implementación. En primer lugar declaramos una serie de constantes a la clase que se muestran en la tabla 7.1:

Tabla 7.1: Constantes de la clase *Util.*

Constante	**Descripción**
LATEX	Identifica salida en Latex
MATHJAX_JSP	Identifica salida en JSP usando MathJax
SIETTE_JSP	Identifica salida en JSP usando Siette
MINIMA_PRECISION	Establecer la precisión mínima a utilizar (nº de decimales)
MIN_NUM_VAR	Establece número mínimo de variables aleatorias en el sistema
MIN_NUM_VAL	Establece número mínimo de valores de una variable aleatoria
MAX_TIPO_PREGUNTAS	Nº máximo tipos de preguntas implementadas en la librería

A continuación vamos a comentar brevemente los métodos de la clase más representativos que nos servirán para cubrir las distintas funciones o acciones que se repiten y necesitamos en varios puntos de la implementación. Para una descripción detallada de las distintas variantes de los métodos (métodos sobrecargados), parámetros y valores de retorno consultar la documentación *Javadoc* generada del *API* de la librería.

El Método *generarConfiguracionesVector()* es clave para la generación aleatoria de una distribución de probabilidad entre las distintas configuraciones posibles que se pueden dar entre las variables aleatorias de un sistema. Es decir para obtener los valores numéricos asociados a cada combinación de manera que se cumpla la definición de probabilidad

Tabla 7.2: Métodos de la clase *Util*.

Método	**Descripción**
getRandomNum()	Genera un número real aleatorio hasta un máximo
getRandomBool()	Genera aleatoriamente un valor booleano
getRandomMOverN()	Genera m números aleatorios entre un total de N sin repetición
ajustar()	Ajusta el valor a la precisión especificada
TexToPdf()	Convierte fichero en .TEX a archivo PDF
SvgToPdf()	Convierte fichero en .SVG a archivo PDF
generarConfiguracionesVector()	Genera las distintas configuraciones posibles que se pueden dar entre los valores de las variables aleatorias de un vector
factorial()	Calcula el factorial de un número real

conjunta 4.1.1 Este método será la base en la generación de preguntas del primer y tercer bloque temático comentado en la introducción a este capítulo: en el primer bloque para obtener las distribuciones de probabilidad que se plantean en los enunciados y en el tercer bloque para obtener los potenciales o distribuciones de probabilidad asociada a cada nodo de la red bayesiana. A continuación vamos a explicar su funcionamiento.

El método realiza todas las combinaciones posibles entre los distintos valores que pueden tomar cada una de las variables y permite obtener la probabilidad, es decir el valor numérico, de cada una de las combinaciones. Este valor numérico se puede obtener de dos maneras:

1) A partir de las probabilidades marginales de los valores de las variables aleatorias que se combinan, probabilidad marginal de cada valor que puede tomar la variable y que a su vez constituye una distribución de probabilidad.

2) A partir de la probabilidad marginal de la combinación sobre una configuración del sistema dada (esa configuración del sistema será recibida como tercer parámetro del método). En la tabla 7.3 se muestran cada uno de los parámetros que puede recibir el método:

El parámetro *vectorConfiguraciones* será un parámetro de salida donde se van a devolver el resultado de las combinaciones realizadas entre los valores del vector de variables aleatorias pasado con su correspondiente valor numérico asociado. El método garantiza que se cumplan al menos el número de relaciones de independencia, establecido en el último parámetro K, entre las variables del vector.

Tabla 7.3: Parámetros del método *generarConfiguracionesVector()*.

Parámetros	**Descripción**
vectorVariables	vector con las variables aleatorias a combinar
vectorConfiguraciones *(parámetro salida)*	resultado de las configuraciones *de las variables VectorVariables*
configuraciones	configuración dada para calcular probabilidad marginal en caso 2)
Ip	indicador de cómo calcular la probabilidad Valores posibles: true→ caso 1), false→: caso 2)
K	nº relaciones de independencia mínimo que se deben cumplir

7.1.2.2. Clase *TextFormat*

Incluye toda la definición de constantes necesarias para formatear la salida de cadenas de texto en función del tipo de salida: consola, *Latex* o *MathJax/JSP*. Esta labor la realiza el método *formatString()*.

7.1.3. Paquete *Siette*

En este paquete definiremos además la jerarquía de interfaces que deberán implementar las clases que plantean las preguntas para su integración en el sistema *Siette*. Toda clase que defina un tipo de pregunta en el sistema *Siette* ha de implementar todos los métodos definidos en los interfaces del apartado 6.1.2. Dependiendo de la pregunta se implementará uno de los siguientes:

- si es de respuesta simple, es decir sólo una respuesta de las alternativas planteadas es correcta, implementará el interfaz *RespuestaSimple*.
- si es de respuesta múltiple, es decir puede haber más de una respuesta correcta entre las alternativas planteadas, en este caso se implementará los métodos de la interfaz *RespuestaMultiple*.

A continuación comentamos cada uno de estos interfaces con sus métodos asociados.

7.1.3.1. Interface *PreguntaSiette*

Establece parte de la interfaz a implementar por cualquier clase que quiera integrarse con el sistema *Siette*. Este interfaz deriva o se especializa en dos interfaces distintos dependiendo del tipo de pregunta que se quiera definir. Una clase que se integra en *Siette* debe implementar uno de los dos interfaces derivados que se explicarán en los siguientes

apartados. Los métodos de este interfaz Java junto con los de sus derivados, son los necesarios para que una clase de la librería implementada pueda ser integrada como parte del sistema *Siette* y ser utilizada para la elaboración de preguntas que formen parte de ejercicios de test o exámenes de autoevaluación que puedan ser realizados por los alumnos y usuarios del citado sistema. En la tabla 7.4 se comentan cada uno de los métodos que presenta este interfaz:

Tabla 7.4: Métodos del interface *PreguntaSiette.*

Métodos	**Descripción**
getEnunciado()	Obtiene el enunciado de la pregunta
getRespuestasIncorrectas()	Devuelve un vector con todas las respuestas incorrectas
getRefuerzoRespuestasIncorrectas()	Devuelve el refuerzo de las respuestas incorrectas

7.1.3.2. Interface *RespuestaSimple*

El interface *RespuestaSimple* hereda del interfaz definido en el apartado anterior incorporando dos nuevos métodos quedando, de esta manera, completa la funcionalidad que ha de aportar una clase que implemente una pregunta de test que sea integrable en el sistema Siette compuesta por el enunciado de una pregunta y una única respuesta correcta y varias incorrectas. En la tabla 7.5 se presentan los métodos de este interfaz:

Tabla 7.5: Métodos del interfaz *RespuestaSimple.*

Métodos	**Descripción**
getRespuestaCorrecta()	Devuelve la respuesta correcta
getRefuerzoRespuestaCorrecta()	Devuelve el refuerzo de la respuesta correcta

7.1.3.3. Interface *RespuestaMultiple*

El tercer y último interfaz, incorpora dos nuevos métodos quedando, de esta manera, completa la funcionalidad que ha de aportar una clase que implemente una pregunta de test que sea integrable en el sistema *Siette* compuesta por el enunciado de una pregunta y una o varias respuestas correctas y varias incorrectas.

Tabla 7.6: Métodos del interfaz *RespuestaMultiple.*

Métodos	**Descripción**
getRespuestasCorrectas()	Devuelve un vector con todas las respuestas correctas
getRefuerzoRespuestasCorrectas()	Devuelve el refuerzo de las respuestas Correctas

7.1.3.4. Clase *Question*

El paquete se completa con la declaración de la clase abstracta *Question* para la generación automática de preguntas. Una pregunta estará formada por un enunciado y una o varias posibles respuestas correctas. Se podrá establecer el número de respuestas posibles que se plantean, así como cuántas de ellas son correctas. Plantea el concepto de una pregunta genérica en la que se plantea una pregunta y se presentan un número determinado de posibles respuestas correctas y otras incorrectas a modo de pregunta típica de test.

De esta clase derivarán todos los tipos de preguntas que se plantean en este proyecto en las que en cada una de ellas se abordará un bloque temático concreto.

En la tabla 7.7 se comentan los atributos definidos en la clase:

Tabla 7.7: Atributos de la clase *Question.*

Atributo	**Descripción**
nCorrectas	número de respuestas correctas a la pregunta planteada
nIncorrectas	número de respuestas incorrectas a la pregunta planteada
orden	orden de presentación de las respuestas elegidas. Este orden se elige de manera aleatoria
nombre	nombre que se le da a la pregunta en cuestión

La clase se completa con los métodos de acceso correspondientes (*getters* y *setters*) a cada uno de los atributos definidos en el cuadro anterior.

7.1.4. Paquete *examen*

El paquete que completa los componentes comunes es el paquete *es.uned.dia.pfc.lgap.mgp.examen* (ver 6.1.3).

En los apartados siguientes comentamos cada una de las clases por separado.

7.1.4.1. Clase *ExamenTest*

La clase *ExamenTest* genera una pregunta o conjunto de preguntas en formato PDF. La pregunta o conjunto de preguntas a generar se pasan como parámetro al constructor y ha de ser un vector de objetos del tipo *Question*. El método *genPDFLatexFile()* generará un archivo en formato Latex cuyo contenido será una o varias preguntas que componen un examen en formato test. Cada pregunta se compone de un enunciado y de las posibles respuestas. Posteriormente se realizará la conversión al formato PDF. Los objetos de tipo *Question* que se pasan como parámetro han de implementar el interface *ExamenTestPdf* que se comenta en el siguiente apartado.

7.1.4.2. Interface *ExamenTestPdf*

Define un examen de test como una colección de una o varias preguntas. El interface presenta el método *genTextFilePreguntas()* que escribirá en el fichero de salida el contenido de cada pregunta que forma parte del examen de test. El método recibe un objeto del tipo *PrinterWriter* en donde se escribirá cada una de las preguntas que forman parte del examen.

7.2. Relaciones de Independencia Probabilística sobre distribución de probabilidad

En este apartado vamos a explicar como se ha realizado la implementación de los tipos de preguntas que se plantean sobre el primer bloque temático que tiene que ver con el planteamiento de preguntas que se definen sobre un sistema que se compone de un conjunto de variables aleatorias y unas relaciones de independencia que se establecen sobre las variables del sistema a partir de una distribución de probabilidad definida sobre los valores que pueden tomar las variables aleatorias que componen el sistema. Empezaremos describiendo las clases que forman el paquete *es.uned.dia.pfc.lgap.mgp.sistema* (ver 6.2.1) y a continuación especificaremos los distintos tipos de pregunta que se pueden plantear sobre el sistema definido.

7.2.1. Paquete *sistema*

En este paquete se agrupan todas las clases que tienen que ver con un sistema como el descrito en el apartado 4.1.1. En el siguiente diagrama se muestran las clases de las

que se compone el paquete *es.uned.dia.pfc.lgap.mgp.sistema* identificando los atributos y métodos más representativos.

En el diagrama anterior, se representan las clases consideradas para la construcción de sistemas definidos como un conjunto de variables aleatorias entre las cuáles se establecen una serie de relaciones de independencia probabilística y de independencia probabilística condicional. Cada clase definida representa un concepto y la suma de todos ellos nos proporciona el sistema completo que contiene todos los elementos necesarios para la generación automática de preguntas de test que tienen en cuenta estos conceptos. Estas clases representan la base teórica y conceptual alrededor de la cuál se plantearan las distintas preguntas que evalúen si el alumno ha asimilado o no algunos conceptos. En los apartados siguientes se comentarán cada una de estas clases por separado.

Todas las clases implementarán y tendrán en común como mínimo los métodos *equals()*, *toString()* y *hashCode()*. Además se implementará el método *toLatexString()* que complementará al método *toString ()* para formatear en *Latex* la cadena que representa el objeto instanciado en la clase.

7.2.1.1. Clase *ValorGenerico*

En esta clase se representa el concepto de valor genérico como la parte más básica del sistema. Se trata de una clase abstracta que nos sirve para englobar los distintos tipos de valores que puede tomar una variable aleatoria y que serán los valores concretos dentro de un rango que hacen que la variable aleatoria quede definida de forma **exclusiva** y **exhaustiva**. Por ejemplo, a la variable aleatoria edad podemos asociarle tres valores: "menor de 18 años", "de 18 a 65 años" y "mayor de 65 años". Estos valores son exclusivos porque son incompatibles entre sí: una persona menor de 18 años no puede tener de 18 a 65 años ni más 65, etc. Son también exhaustivos porque cubren todas las posibilidades. En vez de escoger tres intervalos de edad, podríamos asignar a la variable edad el número de años que tiene la persona; en este caso tendríamos una variable numérica. Cuando el rango o conjunto de valores que puede tomar una variable presenta cardinalidad dos, particularizamos un poco más y hablamos de valores binarios y de variables aleatorias binarias como aquellas que única y exclusivamente pueden tomar un valor de entre dos posibles en un momento dado. Estas consideraciones nos llevan a la definición de las siguientes clases derivadas de la clase abstracta *ValorGenerico* y que especializan más aún el concepto de valor que puede tomar una variable aleatoria: *ValorConcreto* y *ValorBinario*.

7.2.1.2. Clase *ValorConcreto*

La clase *ValorConcreto* representa un valor concreto que puede tomar una determinada variable aleatoria dentro del sistema. Por ejemplo "menor de 18 años" para la variable aleatoria Edad, "mujer" para la variable aleatoria Sexo, etc. El concepto "valor concreto", viene representado en la clase por un nombre (que será un tipo *String* de Java) que identifica al valor concreto y la clase dotará de los métodos de acceso y modificación adecuados a dicho atributo (*get* y *set*).

7.2.1.3. Clase *ValorBinario*

Representa un valor concreto binario. Una variable aleatoria binaria del sistema sólo podrá tomar un valor de entre dos posibles. Por ejemplo la variable binaria X puede tomar los dos únicos valores $+x$ o $\neg x$. Esta clase hereda de la clase *ValorConcreto* anterior e incorpora la propiedad o atributo "valor" para indicar la presencia o ausencia de una determinada propiedad en el concepto que representa la variable aleatoria. Por tanto vamos a pensar en variables aleatorias binarias que presentan o no una determinada propiedad y que representaremos mediante los símbolos "+" para la presencia y "¬" para la ausencia de esa propiedad.

7.2.1.4. Clase *ConjuntoValores*

Con ella se representará una agrupación de valores genéricos a los que opcionalmente se les puede asociar un nombre. Esta clase será la base para la definición del concepto de variable aleatoria discreta así como del concepto de configuración, ambos representados como un conjunto de valores y que se detallan en los siguientes apartados. En el caso de las variables aleatorias, asociaremos a cada valor que pueda tomar la variable una probabilidad, la probabilidad de que la variable tome cada valor, se decir probabilidad marginal (ver 4.1.1). Para el caso de las configuraciones, asociaremos una única probabilidad al conjunto de valores representado, la probabilidad de que se presente esa combinación (configuración) determinada en el sistema, es decir la probabilidad conjunta (ver 4.1.1). Si la variable aleatoria es binaria el conjunto de valores estará formado por dos objetos de la clase *valorBinario* mientras que si es una variable no binaria, estará formada por 'n' objetos de la clase *valorConcreto*.

Tendremos pues en la clase dos atributos: un *String* para representar el nombre del conjunto y un vector que representa el conjunto como una colección de valores genéricos.

Además de los métodos de acceso y modificación (*getters* y *setters*) los métodos *toS-*

tring(), *toLatexString()* y *hashCode()* implementaremos el método *constains()* para cada tipo de valores de los que pueda estar formado el conjunto. Estos métodos nos servirán de base para la correcta implementación del método *equals()*.

7.2.1.5. Clase *Configuracion*

La clase *Configuracion* constituye una agrupación de valores, es decir, un conjunto formado por un valor de cada una de las variables aleatorias del sistema o de parte de ellas en cuyo caso estaríamos hablando de una proyección. Especializa el concepto de conjunto de valores y por tanto hereda de la clase comentada en el apartado anterior *ConjuntoValores*.

Incorpora dos atributos que son la probabilidad, asociada a la configuración, y la precisión que será el número de decimales con los que se representa la citada probabilidad. Dicha probabilidad es la probabilidad de que esa configuración se dé en el sistema, es decir, de que las variables aleatorias, tomen cada una el valor correspondiente de la configuración, es como ya hemos comentado lo que llamamos la probabilidad conjunta.

7.2.1.6. Clase *DistribucionP*

Clase que representa una distribución de probabilidad, definida como un conjunto de configuraciones de los valores de las variables aleatorias. Constituirá un conjunto de elementos del tipo *Configuracion.* Opcionalmente se le puede asociar un nombre.

7.2.1.7. Clase *Probabilidades*

La clase *Probabilidades*, nos será util siempre que tengamos que trabajar con un conjunto de número reales, representando en el contexto de este desarrollo, el concepto de probabilidad. Es el caso de la definición de una variable aleatoria como definimos en el apartado siguiente.

7.2.1.8. Clase *VarAleatoria*

Es la clase principal del paquete sistema y representa el concepto de variable aleatoria como un conjunto discreto de valores que definen a la variable de forma exclusiva y exhaustiva. Cada valor llevará asociado un número real que representa la probabilidad de que la variable aleatoria tome ese determinado valor. Por este motivo tendremos un atributo que será un objeto de la clase *Probabilidades* donde almacenaremos la probabilidad asociada a cada valor que puede tomar la variable aleatoria.

En esta clase destacamos el método *genProbabilidades()*. Este método será el encargado de establecer aleatoriamente las probabilidades de cada uno de los valores en el momento en que se construye el objeto de variable aleatoria o cada vez que se incorpora un nuevo valor a la misma. La suma de las probabilidades de todos los valores que puede tomar una variable aleatoria debe ser la unidad al tratarse de una distribución de probabilidad. Dicho método nos permitirá modificar o generar nuevas probabilidades asociadas a cada valor. Este método nos será útil posteriormente para exigir que se cumplan las propiedades de independencia probabilística entre las variables del sistema.

7.2.1.9. Clase *VarBinaria*

Se trata de un caso especial de variable aleatoria que tiene la particularidad de que sólo puede tomar un valor de entre dos posibles: uno de estos valores representando el concepto de presencia de una determinada propiedad y el otro la ausencia de la misma. Por tanto, la clase *VarBinaria* heredará de la clase *VarAleatoria* y será una variable formada por dos únicos valores que se reprensentarán con el nombre que de la variable binaria que se pase en el constructor en minúsculas anteponiendo a dicho nombre el carácter "+" para el primer valor y el carácter "¬" para el segundo valor que puede tomar la variable (estos caracteres están definidos como constantes dentro de la propia clase *ValorBinario* para la representación del valor).

7.2.1.10. Clase *Relacion*

Esta clase abstracta agrupará los dos tipos de relación que se van a considerar en el sistema: la relación de independencia probabilística y la relación de independencia probabilística condicional. Las clases derivadas de estas tendrán que implementar el método *checkIp()* que realizará las comprobaciones oportunas, en cada caso, para determinar si se da o no la relación entre las variables aleatorias.

7.2.1.11. Clase *RelacionIp*

Clase que representa el concepto de relación de independencia probabilística entre variables aleatorias. La clase estará constituida por un conjunto de variables aleatorias que son las que constituyen la relación de independencia probabilística o no.

En esta clase podemos destacar el método *CheckIp()* donde se comprueba si las variables aleatorias de la relación cumplen la propiedad de independencia probabilística sobre una configuración dada del sistema, de acuerdo a la definición dada en 4.1.1. El método

recibe como parámetros la configuración del sistema sobre la que se comprobará si se verifica o no la propiedad.

Además el método *getRefuerzoCheckIp()* nos servirá para determinar el refuerzo que justifique el porqué se cumple o no se cumple la propiedad de independencia probabilística representada en la clase.

7.2.1.12. Clase *RelacionIpCond*

De manera análoga al caso anterior, la clase representa el concepto de relación de independencia probabilística condicional entre dos grupos de variables aleatorias. La clase estará constituida por un conjunto de variables aleatorias que son las que constituyen la parte izquierda de la relación y de otro conjunto que constituye la parte derecha (es decir la parte izquierda o derecha de la barra '|' que se utiliza en la notación $I_p(X, Y \mid Z)$). La relación de dependencia se da entre los dos conjuntos.

A destacar el método *CheckIpCond()* donde se comprueba si las variables aleatorias de ambos conjuntos de la relación cumplen la propiedad de independencia probabilista condicional sobre una configuración del sistema dada, de acuerdo a la definición dada en 4.1.1 El método recibe como parámetros la configuración del sistema sobre la que se comprobará si se verifica o no la propiedad

El método *getRefuerzoCheckIp()* nos servirá para determinar el refuerzo que justifique el porqué se cumple o no la propiedad de independencia probabilística condicional.

7.2.1.13. Clase *Sistema*

Clase que engloba a todas las demás y que representa un Sistema completo, es decir un modelo matemático de una porción del mundo real establecido como un conjunto de variables aleatorias (discretas). En el sistema se determinarán los posibles valores de cada una de ellas, estos valores han de ser exclusivos y exhaustivos. Además tendremos una determinada configuración del sistema que vendrá dada por una distribución de probabilidad que asocia a cada combinación entre los valores que pueden tomar las variables un valor numérico que representa la probabilidad conjunta de que las variables tomen los valores dados en la combinación. El sistema con el que vamos a trabajar queda definido pues como un conjunto de variables aleatorias (vector de tamaño N de objetos del tipo de la clase *VarAleatoria*) y una distribución de probabilidad (objeto de la clase *DistribucionP*).

Por lo tanto nuestro sistema estará compuesto por dos vectores:

- un vector de variables aleatorias. 7.2.1.8

- una distribución de probabilidad. 7.2.1.6

Dentro de la clase *sistema*, además de los métodos *getters* y *setters* para acceder y modificar los distintos componentes del sistema, destacamos los métodos siguientes:

- *generarConfiguraciones()*: Genera las distintas configuraciones posibles que se pueden dar entre los valores de las variables aleatorias que componen el sistema. Además se fuerza el cumplimiento de la propiedad de dependencia condicional con respecto de alguna variable del sistema cuyo índice se recibe como parámetro. El algoritmo elegido garantiza que se cumplirá como mínimo k propiedades de independencia probabilística condicional. Las probabilidades asociadas a cada configuración se obtienen forzando a que se cumplan al menos 'K' relaciones de independencia probabilística condicional respecto a la variable *'nVar'* indicada. Se basa en el método *generarConfiguracionesVector()* de la clase *Util* 7.1.2.1

Métodos para imprimir los distintos componentes del sistema:

Todos los métodos que se comentarán a continuación, nos permitirán generar la salida de texto correspondiente para la generación de las preguntas de test planteadas en los apartados anteriores. La salida puede ser texto plano para salida a consola, o puede estar formateada como texto en Latex para la generación en formato PDF o formateada como texto en MathJax/JSP para su integración con el sistema Siette.

- *imprimeVar()*: Imprime las distintas variables del sistema, intercalando el separador dado. Las variables se pueden imprimir sólo nombrándolas o especificando también los valores que pueden tomar. El primer parámetro del método indica el modo en que se visualizará la variable. El último parámetro indica el tipo de salida: consola, *Latex* o *MathJax/JSP*. Por ejemplo ante la llamada *imprimeVar(true, ";", 0)* obtendremos:

$$A(+a, \neg a);\ B(+b, \neg b);\ C(+c, \neg c)$$

- *imprimeProbabilidades()*: Imprime las probabilidades de las variables del sistema (probabilidades marginales de cada uno de los valores). Ejemplo:

$$P(+a) = 0{,}18$$
$$P(\neg a) = 0{,}82$$
$$P(+b) = 0{,}47$$
$$P(\neg b) = 0{,}53$$
$$P(+c) = 0{,}28$$
$$P(\neg c) = 0{,}72$$

- *imprimeConfiguraciones()*: Imprime las distintas configuraciones que se pueden dar en el sistema. Imprimirá una tabla con todas las combinaciones posibles que se pueden dar entre todos los valores de las variables del sistema con su probabilidad conjunta correspondiente. Es decir una distribución de probabilidad que asocia a cada combinación (configuración) un valor numérico que representa la probabilidad de que se dé esa configuración.

Con la definición de esta clase tendríamos la base para definir posteriormente lo que constituye la parte cuantitativa de una red bayesiana.

7.2.2. Paquete *relipDP*

En este punto comentamos las clases que se incorporan al paquete *es.uned.dia.pfc.lgap.mgp.relipDP* y que tienen que ver con las relaciones de independencia probabilística y de independencia probabilística condicional sobre una distribución de probabilidad. En la figura 6.7 representamos el diagrama de clases del paquete:

En los apartados que siguen a continuación, comentamos cada una de las clases que forman parte del paquete.

7.2.2.1. Clase abstracta *IpRelQuestDP*

Clase abstracta para la generación automática de preguntas que tienen que ver con las relaciones de independencia probabilística y de independencia probabilística condicional entre las variables aleatorias de un sistema. La clase implementa el interfaz *ExamenTestPdf* para la generación de cada pregunta formando parte del examen en salida PDF. El sistema está compuesto por un conjunto de variables aleatorias y una distribución de probabilidad. Lo más interesante a comentar en esta clase son los atributos que la definen y que serán heredados por las clases derivadas. Los atributos son los siguientes:

system Definición del sistema. Se declarará un objeto de tipo *Sistema*.

ipRel Para representar las relaciones de independencia probabilística entre variables aleatorias se declara un vector que contendrá todas las relaciones posibles entre dos variables aleatorias del sistema que pueden tener una relación de independencia probabilística 7.2.1.11.

ipRelCond Las relaciones de independencia probabilística condicional entre variables aleatorias se representarán mediante un vector que contiene todas las relaciones

entre variables aleatorias del sistema posibles entre las que puede establecerse una relación de independencia probabilística condicional 7.2.1.12.

En cuanto a los métodos proporcionados por la clase comentar lo siguiente:

- La parametrización del constructor nos permitirá definir las distintas instancias de la clase que darán lugar a preguntas distintas o variaciones sobre el mismo tipo de preguntas tal como se definió en el apartado 6.2.1. El constructor de la clase recibirá como parámetros la precisión a utilizar (número de decimales), el número de variables aleatorias de las que se compondrá el sistema y el tipo de las variables. El tipo puede tomar tres posibles valores:

  - Tipo 1: Sistema compuesto única y exclusivamente por variables aleatorias binarias (nombradas como A, B,C,...).
  - Tipo 2: Sistema compuesto única y exclusivamente por variables aleatorias no binarias (nombradas como V,W, X,...).
  - Tipo 3 : Sistema compuesto tanto por variables binarias (nombradas como A, B,C,...) como por no binarias (nombradas como V,W, X„...). Se elegirá al azar si se crea un objeto de tipo variable binaria o variable aleatoria.

En la implementación del constructor de la clase, tras la declaración de un objeto de la clase *Sistema*, se crearán en función del tipo pasado, los objetos correspondientes, ya sean de tipo *VarBinaria*, *VarAleatoria* o una combinación aleatoria de ambas. Los objetos de tipo *VarBinaria* se compondrán de dos valores binarios (nombrados con el nombre de la variable en minúscula anteponiendo el símbolo '+' o '¬' según corresponda) mientras que para los objetos de tipo *VarAleatoria* se elegirá al azar el número de valores que puede tomar la variable (nombrados con el nombre de la variable a la que pertenece el valor en minúscula y con un subíndice).

Una vez se han definido todas las variables de las que se compone el sistema, pasaremos a generar las distintas configuraciones del mismo, es decir las distintas combinaciones entre los distintos valores que pueden tomar cada una de las variables aleatorias del sistema. El método *generarConfiguraciones()* de la clase *Sistema* se encargará de ello como se ha comentado anteriormente (ver 7.2.1.13).

Para finalizar, en el constructor se creará el vector para las relaciones de independencia probabilística así como el vector para las relaciones de independencia probabilística condicional. Una llamada al método *buildRelacionesIp()* obtiene las distintas combinaciones

entre las variables del sistema para establecer las posibles relaciones de independencia probabilística y de independencia probabilística condicional para la correspondiente llamada a *buildRelacionesIpCond()*.

- Métodos de acceso y modificación (*getters* y *setters*) a los vectores que representan las relaciones probabilística de independencia y de independencia condicional. Nos permitirán plantear las distintos apartados a la pregunta realizada.

En los apartados siguientes vamos a comentar las clases que derivan de *IpRelQuestDP* y que van a dar lugar a la definición de los distintos tipos de pregunta que se definen sobre el sistema que se plantea en la citada clase.

7.2.2.2. Definición de clase para la pregunta tipo 1.- *IpRelQuestion1*

Clase para la generación automática de preguntas de tipo 1.(ver 6.2.1.1).

Esta clase hereda de la clase *IpRelQuestDP* explicada en el apartado anterior e implementa el interfaz *RespuestaMultiple*. En este tipo de preguntas, podemos generar instancias de preguntas con una sola respuesta o que admitan más de una respuesta correcta para la pregunta planteada.

La clase dispondrá de los siguientes atributos:

- Un vector con las respuestas correctas, que contendrá todas las relaciones entre variables, ya sean de independencia probabilística como de independencia probabilística condicional que hacen cierta la expresión, es decir que hacen que se cumpla la relación expresada entre las variables. Este vector contendrá objetos del tipo *Relacion*.

- Un vector con las respuestas incorrectas, donde tendremos las relaciones entre variables cuya propiedad expresada no se cumple, es decir es falsa. Al igual que antes los objetos serán del tipo *Relacion*. En cada caso se especializará en objetos de tipo *RelacionIp* o *RelacionIpCond* según sea la relación expresada entre las variables.

- Un vector con las respuestas elegidas para plantear la pregunta a partir del vector de respuestas correctas y el de incorrectas.

Con los atributos definidos en los puntos anteriores, se pueden plantear distintas instancias de preguntas del tipo 1 variando el número de respuestas planteadas y cuantas de ellas son correctas e incorrectas. De esta manera pondremos a disposición del usuario Siette con el rol de profesor varias posibilidades a la hora de plantear las preguntas.

Métodos más representativos:

Constructor: Recibe la información suficiente para construir el sistema (precisión, nº de variables y tipo) así como el número de respuestas (alternativas) que se van a plantear (*nRespuestas*) y cuántas de las cuáles van a ser correctas (*nRespCorrectas*). Se fuerza la relación de independencia condicional con respecto a la variable indicada en *nVarDependiente* (nº de orden que ocupa la variable en el sistema), esta variable es elegida al azar.

En primer lugar se invoca al constructor de la clase base estableciendo la precisión, el número de variables de las que se compone nuestros sistema, así como el tipo de variables (binarias, aleatorias en general o una mezcla de ambas). A continuación se generan las distintas configuraciones entre las variables aleatorias del sistema. En la llamada al método de la clase *Sistema* le pasamos los parámetros correspondientes de manera que se garantice que tendremos al menos un número determinado de respuestas correctas, es decir de relaciones Ip que se verificarán en la distribución de probabilidad generada y que ésta estará condicionada a una variable determinada. Una vez establecidas el número de alternativas, respuestas que se van a plantear, determinamos el orden en el que se van a plantear las respuestas. mediante el método de la clase base *setOrden(NRespPlanteadas)*

El último paso será chequear las relaciones establecidas en el punto anterior para determinar si se cumplen o no las propiedades indicadas y establecer si las respuestas son correctas o incorrectas y almacenando el resultado en el vector correspondiente de respuestas correctas o incorrectas respectivamente. Los métodos *getRespuestasCorrectas()* y *getRespuestasIncorrectas()* consultarán el vector adecuado para devolver la cadena de texto con el resultado adecuado.

- *imprimeRespuestas()*: Imprime las respuestas planteadas en el orden establecido.. Parámetros:
  - 1 a 1 respuesta: flag indicador de si se marca o no la respuesta correcta.
  - *tipoSalida*: tipo de salida consola, Latex o MathJax (ver 7.1.2.1).
- *getEnunciado()* obtiene el enunciado de la pregunta que se plantea. Este método será invocado desde el apartado correspondiente de la definición de la pregunta en el sistema Siette.
- *getSolucion()* obtendrá la solución a la pregunta planteada con la información de refuerzo necesaria para explicar como se ha llegado a la solución.

- *genTextFilePreguntas()*: implementación del método correspondiente al interfaz *ExamenTestPdf*. Genera ejercicio con salida a archivo Latex .TEX para su posterior conversión a PDF haciendo uso del método correspondiente (ver 7.1.2.1) de la clase *Util*.

- La clase implementa además los métodos del interface *PreguntaSiette* para su integración en Siette que son los siguientes:

7.2.2.3. Definición de clase para la pregunta tipo 2.- *IpRelQuestion2*

Clase para la generación automática de preguntas de tipo 2 (ver 6.2.1.2).

Al igual que en el caso anterior, la clase hereda de la clase *IpRelQuestDP* e implementa el interfaz *RespuestaMultiple* de manera que podemos generar instancias de preguntas con una sola respuesta o que admitan más de una respuesta correcta para la pregunta planteada.

En este caso la clase dispondrá de los mismos atributos que la clase anterior pero cambiando sus tipos. Las respuestas, en lugar de ser relaciones de independencia (vector de objetos de las clases derivadas de la clase *Relacion* 7.2.1.117.2.1.12), serán distribuciones de probabilidad (vector de objetos de la clase *DistribucionP* 7.2.1.6).

La clase dispondrá de los siguientes atributos:

- Un vector con las respuestas correctas, que contendrá todas las distribuciones de probabilidad que verifican la propiedad de independencia probabilística condicional dada en el enunciado. Este vector contendrá objetos del tipo *DistribucionP*.

- Un vector con las respuestas incorrectas, donde tendremos las distribuciones de probabilidad que no verifican la propiedad de independencia probabilística condicional dada en el enunciado. Al igual que antes los objetos serán del tipo *DistribucionP*.

- Un vector con las respuestas elegidas para plantear la pregunta a partir del vector de respuestas correctas y el de incorrectas.

- Además tendremos un atributo que será un objeto de la clase *RelacionIpCond* para plantear la propiedad de independencia probabilística condicional dada en el enunciado.

Con los atributos definidos en los puntos anteriores, se pueden plantear distintas instancias de preguntas del tipo 2 variando el número de respuestas planteadas y cuantas de ellas son correctas e incorrectas.

Métodos más representativos:

Constructor: Recibe la información suficiente para construir el sistema (precisión, nº de variables y tipo) así como el número de respuestas (alternativas) que se van a plantear (*nRespuestas*) y cuántas de las cuáles van a ser correctas (*nRespCorrectas*). Se fuerza la relación de independencia condicional con respecto a la variable indicada en *nVarDependiente* (nº de orden que ocupa la variable en el sistema) elegida de manera aleatoria.

En primer lugar se invoca al constructor de la clase base estableciendo la precisión, el número de variables de las que se compone nuestros sistema, así como el tipo de variables (binarias, aleatorias en general o una mezcla de ambas).

genRespuestas(): Genera las respuestas correctas e incorrectas que se plantean en la pregunta. Una respuesta correcta será una distribución de probabilidad que verifica la propiedad dada en el enunciado. Para generar una respuesta correcta, se generan las distintas configuraciones entre las variables aleatorias del sistema, mediante la llamada al método de la clase *Sistema* le pasamos los parámetros correspondientes de manera que se garantice que tendremos al menos que se cumple la propiedad indicada y posiblemente alguna otra. Una respuesta incorrecta sería una distribución de probabilidad que se genera forzando a que la distribución de probabilidad cumpla la relación de independencia probabilística condicional respecto de cualquier otra variable que no sea la propuesta en el enunciado, dicha variable se elegirá de manera aleatoria.

A la hora de generar la respuesta, el método comprueba que cada distribución de probabilidad generada sea diferente a las generadas con anterioridad, es decir que no existan duplicados.

- *imprimeRespuestas()*: Imprime las respuestas planteadas en el orden establecido.. Parámetros:
  - 1 a 1 respuesta: flag indicador de si se marca o no la respuesta correcta.
  - tipoSalida: tipo de salida consola, Latex o MathJax (ver 7.1.2.1).
- *getEnunciado()* obtiene el enunciado de la pregunta que se plantea. Este método será invocado desde el apartado correspondiente de la definición de la pregunta en el sistema Siette.
- *getSolucion()* obtendrá la solución a la pregunta planteada con la información de refuerzo necesaria para explicar como se ha llegado a la solución.

- *genTextFilePreguntas()*: implementación del método correspondiente al interfaz *ExamenTestPdf*. Genera ejercicio con salida a archivo latex .TEX para su posterior conversión a PDF haciendo uso del método correspondiente (ver 7.1.2.1).

- La clase implementa además los métodos del interface *PreguntaSiette* para su integración en *Siette* que son los que se muestran en la tabla 7.4

7.2.2.4. Definición de clase para la pregunta tipo 3.- *IpRelQuestion3*

Clase para la generación automática de preguntas de tipo 3. (ver 6.2.1.3)

Esta clase hereda de la clase *IpRelQuestion2* al ser una variación de las preguntas del tipo 2 en las que las distribuciones de probabilidad dadas han de verificar todas las relaciones posibles entre las variables del sistema en lugar de una sola. Además implementa el interfaz *RespuestaMultiple*. En este tipo de preguntas, podemos generar instancias de preguntas con una sola respuesta o que admitan más de una respuesta correcta para la pregunta planteada.

En le método *genRespuestas()* se generan las posibles respuestas que se plantean a la pregunta. Una respuesta correcta será una distribución de probabilidad que verifica todas las relaciones posibles que se pueden dar entre las variables del sistema. Para generar una respuesta correcta, se generan las distintas configuraciones entre las variables aleatorias del sistema, mediante la llamada al método de la clase *Sistema* le pasamos los parámetros correspondientes de manera que se garantice que se cumplen todas las relaciones posibles. Una respuesta incorrecta sería una distribución de probabilidad que se genera forzando a que la distribución de probabilidad cumpla la relación de independencia probabilística condicional respecto de una sola variable, dicha variable se elegirá de manera aleatoria.

A la hora de generar la respuesta, el método comprueba que cada distribución de probabilidad generada sea diferente a las generadas con anterioridad, es decir que no existan duplicados.

El resto de métodos de esta clase se heredan de la clase base.

7.2.2.5. Definición de clase para la pregunta tipo 4.- *IpRelQuestion4*

Clase para la generación automática de preguntas de tipo 4. (ver 6.2.1.4)

Esta clase hereda de la clase *IpRelQuestion2* al ser una variación de las preguntas del tipo 2 en las que las distribuciones de probabilidad dadas han de verificar una determinada propiedad dada en el enunciado. Además implementa el interfaz *RespuestaSimple*. En este tipo de preguntas, podemos generar instancias de preguntas con una sola respuesta.

Para el caso de esta pregunta, el alumno deberá realizar los cálculos oportunos para determinar el valor de las incógnitas planteadas en las filas de la distribución de probabilidad dada de manera que se cumpla la relación de dependencia dada en el enunciado. La respuesta, en esta caso, no consiste en seleccionar una respuesta correcta entre varias alternativas sino en una entrada libre que asocie a cada incógnita planteada el valor numérico que le corresponda para que se cumpla la relación planteada.

En le método *genRespuestas()* elige las filas que se van a mostrar como una incógnita para que el alumno determine cuál es el valor asociado a esa distribución conjunta de la configuración en cuestión.

El resto de métodos de esta clase se heredan de la clase base.

7.3. Relaciones de Independencia Probabilística sobre grafos

En este apartado vamos a explicar como se ha realizado la implementación de los distintos tipos de preguntas que se plantean sobre el segundo bloque temático que tiene que ver con el planteamiento de preguntas definidas sobre un sistema que se compone de un grafo (dirigido o no dirigido) sobre el que se plantean relaciones de independencia probabilística.

7.3.1. Paquete *grafos*

Según la definición de grafo dada en 4.2.1, un grafo es un conjunto de nodos y enlaces. Por tanto los conceptos básicos que se representan en este paquete son los componentes básicos de un grafo. En el diagrama de clases 6.8 se identifican cada uno de estos componentes, definiendo un grafo como un camino que se define como un conjunto de nodos. Además existen una serie de enlace o aristas y cada enlace está formado por dos nodos que son los que enlazan.

En los siguientes apartados explicamos las particularidades de cada una de las clases identificadas en el diagrama de la figura anterior.

7.3.1.1. Clase *Nodo*

Esta clase representa el concepto de nodo de un grafo. Un nodo vendrá modelado en el sistema como una etiqueta de texto. La nomenclatura elegida para los nodos serán las

primeras letras del abecedarios A, B,C,... en otro casos utilizaremos las finales X, Y, Z...En cualquier caso serán letras mayúsculas, normalmente una sóla, aunque podrían ser varias.

Un aspecto muy importante a tener en cuenta será la representación gráfica del grafo que nos permitirá determinar si se cumplen o no las relaciones de independencia probabilística que se van a plantear sobre los nodos de un grafo, estudiando los distintos caminos activos que existen entre un par de nodos dados. Por tanto esta clase dispondrá de atributos que nos permitan representar el nodo en una posición determinada de un plano. Tendrá por tanto unas coordenadas x e y y se representarán con su etiqueta encerrado en un circulo.

Asimismo los atributos *'visitado'* y *'expandido'* nos permitirán implementar los algoritmos de recorrido sobre un grafo para determinar los caminos activos que existen entre dos nodos dados.

La clase incorpora los métodos *getter* y *setter* para el acceso y modificación a los distintos atributos. Destacamos el método *toSvgString()* que nos devolverá una cadena de texto en formato SVG[1] para la representación gráfica del nodo en pantalla.

7.3.1.2. Clase *Enlace*

Esta clase representa el concepto de enlace o arista entre un par de nodos. Se representa como una relación de composición. Un enlace o arista está compuesto por un nodo enlace y un nodo destino. El enlace puede ser de dos tipos:

- **no dirigido** el enlace entre los dos nodos es uno y no se tiene en cuenta el sentido del mismo. Es decir, el enlace entre X e Y es el mismo que el de Y a X. Se representa $X - Y$. En ambos casos un único objeto representa el enlace.

- **dirigido** se tiene en cuenta el sentido del enlace y se considera que la arista o enlace de X a Y es distinto del de Y a X. Se representa mediante una flecha que determina el sentido del enlace. Así en enlace de X a Y se representa $X \rightarrow Y$ o$(Y \leftarrow X)$ mientras que el enlace de Y a X se representa por $X \leftarrow Y$ o $(Y \rightarrow X)$. Ambos tipos de enlace corresponden a objetos distintos.

Al igual que en el caso del nodo, destacamos el método *toSvgString()* que devuelve la cadena de caracteres en formato SVG que representa la arista o enlace que une dos nodos de un grafo. En función de si el grafo es o no dirigido, representaremos las aristas como líneas (no dirigidos) o como flechas (dirigidos).

[1]Lenguaje de gráficos vectoriales

7.3.1.3. Clase *Camino*

Definimos la clase *Camino* como una sucesión de nodos dentro de un grafo. La definición de esta clase nos va a ser útil en la implementación de los algoritmos que calculan los caminos activos entre dos nodos de un grafo. Definiremos un camino como un vector de nodos. La clase incorpora lo métodos de accesos a los nodos del camino y los métodos típicos sobre un vector que nos permiten añadir nodos al camino, eliminarlos, etc.

7.3.1.4. Clase *Grafo*

Una vez hemos definido los componente básicos de un grafo (nodos y enlaces), tenemos ya los elementos necesarios para poder definir la clase *Grafo* como una colección de nodos y enlaces. Se da, por tanto, una relación de composición. Dentro de la clase vamos a considerar para la representación del grafo como una herramienta muy util en los distintos algoritmos que vamos a emplear para la construcción y recorrido sobre grafos la matriz de adyacencia. La matriz de adyacencia, es una de las formas más extendidas de representación de grafos en la memoria de un ordenador. Se trata de una matriz donde se representan los enlaces del grafo en cada fila y columna. Para un grafo de n nodos tendremos una matriz $n \times n$ donde representamos los nodos en las filas y las columnas de la matriz. Si existe un enlace entre dos nodos, la correspondiente fila/columna de esos nodos tendrá el valor 1 o *true* mientras que si los nodos no están conectados tendremos el valor 0 o $false$. Para el caso de los grafos dirigidos la matriz de adyacencia que lo representa es simétrica.

Dos aspectos importantes que vamos a considerar en la representación del grafo son la profundidad y la amplitud.

- **Profundidad** lo definimos como el número de enlaces que hay que recorrer desde el nodo inicial de la representación según avanzamos en el recorrido en profundidad a partir de los descendientes.

- **Amplitud**. será el número de subgrafos cuando estamos ante un grafo no conexo. La amplitud será 1 si el grafo es conexo y se corresponderá con el número de subgrafos. Un parámetro a considerar en el constructor de la clase será el indicador de conexo que activado forzará a que la generación del grafo sea de un grafo conexo.

Aprovechando las características del lenguaje de programación Java y del paradigma de programación *OO*, esta clase abstracta definirá los métodos comunes tanto a grafos dirigidos como no dirigidos. Cuando las acciones implementadas en el método sean comunes a

ambos tipos de grafo se implementará el método en cuestión en la propia clase abstracta. Si las acciones que se implementan en el método difieren el función del tipo de grafo, se delegará a la clase derivada (*GrafoGNA* o *GrafoGDA* que se verá posteriormente) la implementación concreta en cada caso.

Métodos abstractos:

Tabla 7.8: Métodos abstractos de la clase *Grafo.*

Método	**Descripción**
GenEnlaces()	Genera aleatoriamente los enlaces del grafo
BuildGrafo()	Construye grafo, a partir de nodos y matriz de adyacencia
GetVecinos()	Devuelve la lista de nodos vecinos o adyacentes al pasado
GetNVecinos()	Devuelve el nº de nodos vecinos o adyacentes al pasado
GetVecinosNoVisitados()	Devuelve lista nodos vecinos al pasado aún no visitados
GetCaminosActivos()	Calcula los caminos activos entre dos nodos dados

En la tabla 7.9, comentamos el resto de métodos implementados en la clase:

Entre el resto de métodos de la clase, destacar el método *toSvgString()* se basa en los métodos del mismo nombre de las clases *Nodo* y *Enlace* para obtener la cadena de caracteres en SVG que nos permite obtener la representación gráfica del grafo.

7.3.1.5. Clase *GrafoGNA*

Representa el concepto de Grafo No dirigido Acíclico (GNA). En un grafo no dirigido el enlace entre dos nodos (x,y) es igual al enlace (y,x).

Tabla 7.9: Métodos de la clase *Grafo* (continuación)

Método	**Descripción**
GetCaminosBloqueados()	Calcula los caminos activos de un grafo entre dos nodos dados y un tercer nodo que puede o no bloquear el camino.
GetCaminosActivos()	Calcula los caminos activos de un grafo entre dos nodos dados y un tercer nodo que puede o no bloquear el camino.
setAmplitudProfundidad()	Calcula amplitud y profundidad del grafo generado
distribuirGrafo()	Distribuye espacialmente los nodos del grafo en el área definida

7.3.1.6. Clase *GrafoGDA*

Representa el concepto de Grafo Dirigido Acíclico (GDA). En un grafo dirigido el enlace entre dos nodos (x,y) es distinto al enlace (y,x).

7.3.2. Paquete *relipGrafo*

En este apartado comentamos las clases que se añaden específicamente para plantear aquellos tipo de preguntas que tienen que ver con relaciones de independencia e independencia probabilística condicional entre variables aleatorias representadas mediante los nodos de un grafo. La figura 6.9 representa el diagrama de clases del paquete:

En los siguientes apartados comentamos cada una de ellas por separado.

7.3.2.1. Clase *IpRelQuestGrafo*

Clase para la generación automática de preguntas que tienen que ver con las relaciones de independencia probabilística y de independencia probabilística condicional entre las variables aleatorias o nodos de un grafo. Esta clase abstracta se especializa en dos clases posibles en función del tipo de grafo que estemos considerando: dirigido o no dirigido. En los siguientes apartados comentamos ambos tipos.

7.3.2.2. Clase *IpRelQuestGNA*

Clase para la generación automática de preguntas que tienen que ver con las relaciones de independencia probabilística y de independencia probabilística condicional entre las variables aleatorias de un grafo no dirigido acíclico (GNA).

7.3.2.3. Clase *IpRelQuestGDA*

Clase para la generación automática de preguntas que tienen que ver con las relaciones de independencia probabilística y de independencia probabilística condicional entre las variables aleatorias de un grafo dirigido acíclico (GDA).

7.3.2.4. Definición de clase para la pregunta tipo 5.- *IpRelQuestion5*

Clase para la generación automática de preguntas de tipo 5. (ver 6.2.2.1)

Esta clase hereda de la clase *IpRelQuestGNA* e implementa el interfaz *RespuestaMultiple*. En este tipo de preguntas, podemos generar instancias de preguntas con una sola respuesta o que admitan más de una respuesta correcta para la pregunta planteada.

El método *checkRespuestas()* genera 'nRespuestas' alternativas. Cada alternativa presenta una relación de independencia probabilística entre dos variables o nodos del grafo.

El resto de métodos y comportamiento vienen heredados de las clases bases.

La pregunta, consistirá en determinar cuál o cuáles de las relaciones planteadas entre las variables o nodos de un grafo se dan realmente. Es decir, determinar si es verdadera o es falsa la relación especificada en cada apartado. Para la resolución de la pregunta, el alumno deberá comprobar, en base a los enlaces que se dan en el grafo, si se cumple o no la relación propuesta teniendo en cuenta los caminos activos que puedan existir entre las variables o nodos del grafo.

La pregunta tendrá las siguientes variantes en función de la parametrización:

- enunciado de la pregunta con una serie de respuestas o alternativas. El alumno deberá indicar para cada alternativa planteada, si es verdadera o falsa. Las relaciones planteadas en cada alternativa serán relaciones de independencia entre dos nodos o variables del grafo dado en el enunciado.

- igual que en el punto anterior, pero en esta ocasión las relaciones de independencia planteada son condicionadas a una tercera variable o nodo del grafo.

- Se plantea como pregunta con repuesta abierta. La pregunta consistirá en indicar los caminos activos que existen entre dos nodos del grafo entre los que se da una relación de independencia probabilística. El alumno deberá indicar los caminos activos y los bloqueados entre dos nodos.

- Pregunta igual a la anterior con la variación de que en este caso la relación se plantea como relación de independencia probabilística condicional de dos variables o nodos del grafo respecto de una tercera variable. El alumno deberá indicar los caminos activos y los caminos bloqueados entre dos nodos condicionado al tercero.

7.3.2.5. Definición de clase para la pregunta tipo 6.- *IpRelQuestion6*

Clase para la generación automática de preguntas de tipo 6. (ver 6.2.2.2)

Esta clase hereda de la clase *IpRelQuestGDA* e implementa el interfaz *RespuestaMultiple*. En este tipo de preguntas, podemos generar instancias de preguntas con una sola respuesta o que admitan más de una respuesta correcta para la pregunta planteada.

El método *checkRespuestas()* genera *'nRespuestas'* alternativas. Cada alternativa presenta una relación de independencia probabilística entre dos variables o nodos del grafo.

El resto de métodos y comportamiento vienen heredados de las clases bases.

La pregunta, consistirá en determinar cuál o cuáles de las relaciones planteadas entre las variables o nodos de un grafo se dan realmente. Es decir, determinar si es verdadera o es falsa la relación especificada en cada apartado. Para la resolución de la pregunta, el alumno deberá comprobar, en base a los enlaces que se dan en el grafo, si se cumple o no la relación propuesta teniendo en cuenta los caminos activos que puedan existir entre las variables o nodos del grafo.

La pregunta tendrá las siguientes variantes en función de la parametrización:

- enunciado de la pregunta con una serie de respuestas o alternativas. El alumno deberá indicar para cada alternativa planteada, si es verdadera o falsa. Las relaciones planteadas en cada alternativa serán relaciones de independencia entre dos nodos o variables del grafo dado en el enunciado.

- igual que en el punto anterior, pero en esta ocasión las relaciones de independencia planteada son condicionadas a una tercera variable o nodo del grafo.

- Se plantea como pregunta con repuesta abierta. La pregunta consistirá en indicar los caminos activos que existen entre dos nodos del grafo entre los que se da una relación de independencia probabilística. El alumno deberá indicar los caminos activos y los bloqueados entre dos nodos.

- Pregunta igual a la anterior con la variación de que en este caso la relación se plantea como relación de independencia probabilística condicional de dos variables o nodos del grafo respecto de una tercera variable. El alumno deberá indicar los caminos activos y los caminos bloqueados entre dos nodos condicionado al tercero.

7.4. Inferencia en Redes Bayesianas

En este último apartado vamos a explicar como se ha realizado la implementación de los tipos de preguntas que se plantean sobre el tercer y último bloque temático que tiene que ver con el planteamiento de preguntas que se definen sobre un sistema que consiste en una red bayesiana sobre las que se plantean una serie de consultas tras la evidencia introducida en la red. En este tercer bloque temático tomamos como base los dos anteriores, el primero de ellos nos proporciona la parte cuantitativa de una red bayesiana, el segundo su parte cualitativa y en este tercer bloque incorporamos la inferencia que será proporcionada por el *API* del programa OpenMarkov ya comentado anteriormente. Pasamos a explicar el desarrollo realizado en este bloque dentro del paquete "*rb*".

7.4.1. Paquete *rb*

Empezaremos describiendo las clases que forman el paquete *es.uned.dia.pfc.lgap.mgp.rb* y a continuación especificaremos los distintos tipos de pregunta que se pueden plantear sobre el sistema definido 6.12.

7.4.1.1. Clase *IBNQuestion*

Al igual que el resto de clases base de cada bloque temático, la clase abstracta *IBNQuestion* hereda de la clase genérica *Question* (ver 7.1.3.4) que proporciona el concepto de pregunta genérica. En la clase *IBNQuestion* se plantearán los tipos de preguntas relativas a la inferencia en redes bayesianas. También implementa el interfaz *ExamenTestPdf* que nos permitirá incorporar los distintos tipos de pregunta definidas en las clases derivadas como parte de un examen de test en PDF.

Como hemos comentado, en este bloque se reúnen los dos anteriores. De esta forma los atributos de la clase *IBNQuestion* serán:

- un objeto de la clase *Sistema* (ver 7.2.1.13) que representa la parte cuantitativa de una red bayesiana. Este objeto nos permitirá definir la factorización de la red como una distribución de probabilidad definida sobre las variables aleatorias o nodos que componen la red.

- un objeto de la clase *GrafoGDA* (ver 7.3.1.6) que representa el grafo que define la estructura gráfica de una red bayesiana y que constituye, por tanto, su parte cualitativa.

- por último un objeto de la clase *ProbNet* proporcionada por el *API* de *OpenMarkov* (ver OpenMarkovDoc) que nos permitirá construir la red bayesiana y a partir de las clases y métodos proporcionados por el *API* de *OpenMarkov* podremos realizar la inferencia sobre la red.

Estos tres tipos de objetos componen los elementos necesarios para poder definir preguntas de test sobre inferencia en redes bayesianas. Todos ellos presentarán los correspondientes métodos de acceso y modificación (*getters* y *setters*).

Además destacamos, dentro de la clase los métodos siguientes 7.10:

Tabla 7.10: Métodos de la clase *IBNQuestion.*

Método	**Descripción**
getPotentials()	Obtiene los potenciales asociados a las variables de la red
getStates()	Obtiene los estados de cada una de las variables de la red
getProbability()	Obtiene probabilidades marginales de las variables de red

7.4.1.2. Clase *IBNQuestRB*

La clase abstracta *IBNQuestRB* hereda de *IBNQuestion*. Esta clase será la base de todas las preguntas que se definan en la clases derivadas sobre inferencia en redes bayesianas y consistirán en la presentación en el enunciado de una red bayesiana, la introducción de uno o varios hallazgos (evidencia) en la red y preguntar por la probabilidad de alguna variable de interés tras la evidencia introducida en la misma. Tanto los hallazgos como las variables que forman parte de la pregunta se elegirán de manera aleatoria, esta tarea será llevada a cabo por el método *genRespuestas()*.

Tanto el enunciado de las preguntas como el refuerzo proporcionado será común a todas las preguntas que se plantean sobre inferencia en redes bayesianas. Por este motivo los métodos *getEnunciado()* y *getRefuerzo()* serán implementados en esta clase y heredados por todas aquellas clases que deriven de ella.

7.4.1.3. Definición de clase para la pregunta tipo 7.- *IBNQuestion7*

Clase para la generación automática de preguntas de tipo 7. (ver 6.2.3.1) .

La pregunta que se plantea será una pregunta abierta en la que se dan como datos la prevalencia de una determinada enfermedad, así como la sensibilidad y la especificidad de un determinado test para esa enfermedad. La pregunta, consistirá en calcular las probabilidades a posteriori o valores predictivos del Test.

En la segunda parte de la pregunta, se introduce un hallazgo en el test y se pregunta por el valor predictivo positivo de la enfermedad.

La Red Bayesiana que se plantea en la pregunta se representa con dos nodos (X, Y) y un enlace $X \rightarrow Y$. El nodo X representa la enfermedad ($Disease$) mientras que el nodo Y representa el *test* para dicha enfermedad. Haciendo uso del interfaz gráfico proporcionado por el programa *OpenMarkov*, podemos realizar una representación gráfica de la red propuesta, introduciendo los valores del enunciado que determinan la tabla de probabilidad condicionada de cada uno de sus nodos. La figura 7.1 muestra el resultado:

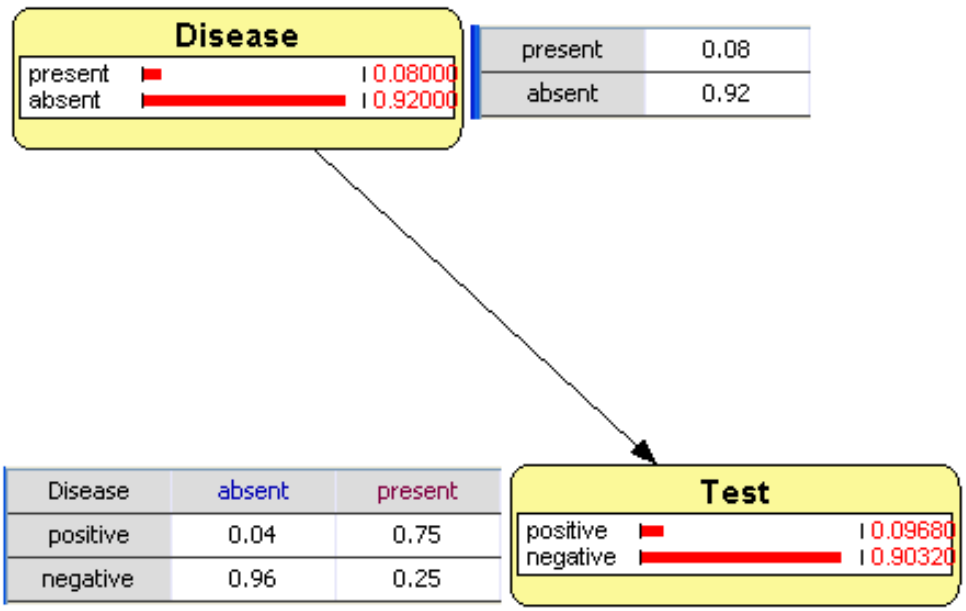


Figura 7.1: Representación de la red bayesiana *Disease/Test* en *OpenMarkov*.

A partir de aquí, si se introduce el hallazgo positivo en el resultado del test (Y = *positive*), el valor predictivo positivo para la enfermedad se representa en la figura 7.2:

En el constructor de la clase *IBNQuestion7* se crearan los tres objetos para la representación de la red bayesiana definidos en 7.4.1.1.

El objeto de la clase *Sistema* se define con dos variables binarias (objetos de la clase *VarBinaria*) que representa los dos nodos de la red, la enfermedad y el test. El objeto de tipo *GraoGDA* se creará con dos nodos, nombrados X e Y que representan a la enfermedad y el test respectivamente y un enlace dirigido entre ambos $X \rightarrow Y$. Por último el objeto de la clase *ProbNet* se construye a partir de la factoría de redes incorporada en el programa *OpenMarkov BN_XY*. La clase de *OpenMarkov* que implementa dicha factoría es *NetsFactory* y el método que crea la red es *createBN_XY()* que recibe como parámetros la prevalencia, la sensibilidad y la especificidad.

Una vez creados todos los elementos necesarios, el resto de métodos de la clase nos permitirán confeccionar la pregunta y se implementa los métodos de la interfaz para su integración con el sistema *Siette*.

7.4.1.4. Definición de clase para la pregunta tipo 8.- *IBNQuestion8*

Clase para la generación automática de preguntas de tipo 8. (ver 6.2.3.2)

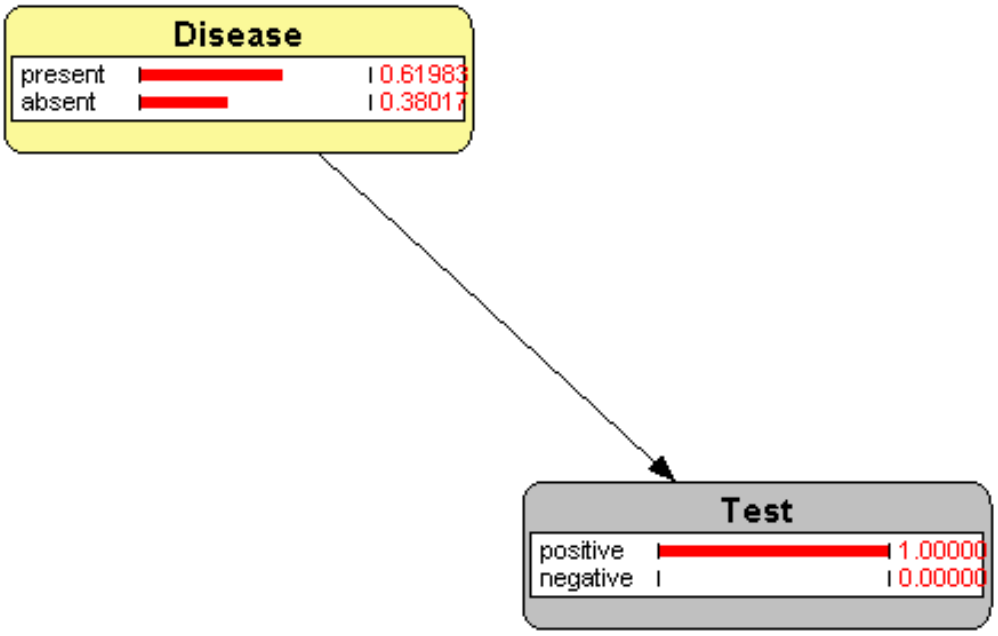


Figura 7.2: Representación de la red bayesiana *Disease/Test* tras la evidencia introducida.

La Red Bayesiana que se plantea en la pregunta será elegida al azar y podrá ser una de las tres ofrecidas por la clase *NetsFactory* de *OpenMarkov*. En el constructor de la clase se llamará al método correspondiente para la creación de la red elegida:

- método *create_bN_ABC()*, creará los tres objetos para la representación de la red bayesiana definidos en 7.4.1.1. El objeto de la clase *Sistema* se define con tres variables binarias (objetos de la clase *VarBinaria*) que representan los tres nodos de la red, A, B y C. El objeto de tipo *GraoGDA* se creará con tres nodos, nombrados A, B y C y tres enlaces dirigidos $A \to B$, $A \to C$ y $B \to C$. Por último el objeto de la clase *ProbNet* se construye a partir de la factoría de redes, el método de la la clase *NetsFactory* de *OpenMarkov* que crea la red es *createBN_ABC()* que no tiene parámetros.

- método *create_bN_XYZ()*, creará los tres objetos para la representación de la red bayesiana definidos en 7.4.1.1. El objeto de la clase *Sistema* se define con tres variables binarias (objetos de la clase *VarBinaria*) que representan los tres nodos de la red, X, Y y Z. El objeto de tipo *GraoGDA* se creará con tres nodos, nombrados X, Y y Z y dos enlaces dirigidos $X \to Y$ y $Y \to Z$. Por último el objeto de la clase *ProbNet* se construye a partir de la factoría de redes, el método de la la clase

NetsFactory de *OpenMarkov* que crea la red es *createBN_XYZ()* que no cinco parámetros, a saber: la prevalencia del nodo X, la sensibilidad y la especificidad del enlace $X \rightarrow Y$, y la sensibilidad y la especificidad relativa al enlace $Y \rightarrow Z$.

- método *create_bN_Asia()*, al igual que en los dos casos anteriores, se creará el objeto de la clase *Sistema* incorporando al mismo los objetos que representan las variables binarias pertenecientes a la conocida red Asia, el grafo se crea nombrando los nodos como $A, S, T, L, B, TOrC, X, C$. En la figura 7.3 observamos la red Asia representada en el interfaz gráfico de *OpenMarkov*.

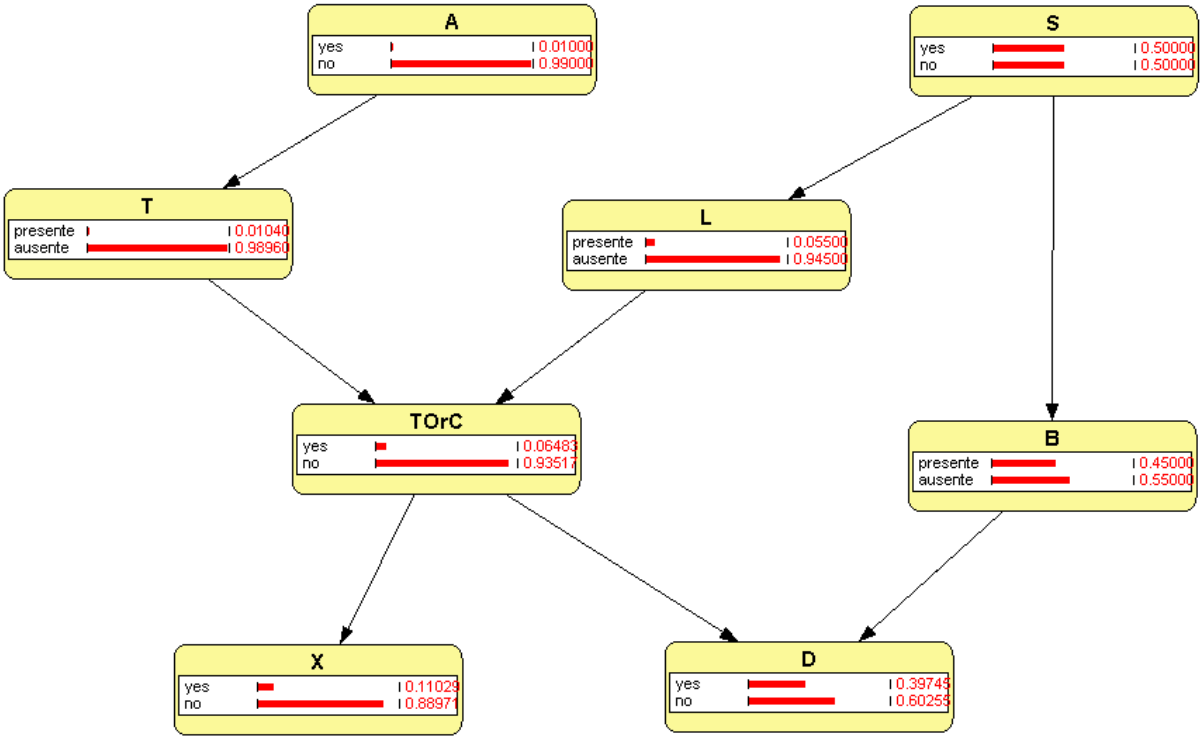


Figura 7.3: Red bayesiana *Asia* en *OpenMarkov*.

Una vez creados todos los elementos necesarios, se hace uso de los métodos heredados de la clase base *IBNQuestRB* para confeccionar la pregunta y su integración con el sistema *Siette*

7.4.1.5. Definición de clase para la pregunta tipo 9.- *IBNQuestion9*

Clase para la generación automática de preguntas de tipo 9. (ver 6.2.3.3).

La red bayesiana que se plantea en este tipo de preguntas se genera aleatoriamente a partir de su representación gráfica.

El grafo GDA asociado a la red se genera en primer lugar, eligiendo sus nodos al azar. A partir del grafo generado, se recorren todos los nodos del mismo generando el potencial (tabla de probabilidad condicionada) asociada a cada nodo de la red. Este potencial se obtiene generando la distribución de probabilidad de un sistema formado por el nodo en cada punto del recorrido y sus antecesores. De esta manera, si un nodo no tiene antecesores, su potencial asociado vendrá dado por la probabilidad marginal asociada a la variable aleatoria del nodo; mientras que si el nodo tiene antecesores, su potencial vendrá dada por la factorización del nodo respecto a sus antecesores según la siguiente expresión (ver 4.3.1):

$$P(x) = \prod_i P_c(x_i \mid pa(X_i))$$

En el constructor de la clase *IBNQuestion9* se generan tantas variables aleatorias binarias (objetos de tipo *varBinaria*) como se indique en el parámetro *nNodos*. Estas variables constituirán el sistema (objeto de la clase *Sistema*) o parte cuantitativa de la red bayesiana. A continuación se generará la parte cualitativa o representación gráfica de la red a través de la creación del objeto de tipo *GrafoGDA*. La nomenclatura utilizada para las variables de la red serán las primeras letras del abecedario en mayúsculas: $A, B, C...$

El parámetro *'genEnlaces'* del constructor de la clase *GrafoGDA*, establecido al valor *true* será el que provoca que se genere el grafo eligiendo los enlaces entre los nodos al azar y el parámetro *'conexo'* exige que el grafo generado sea conexo, requisito indispensable en la construcción de una red bayesiana.

Como se ha comentado, a partir del grafo generado, se va a establecer la parte cuantitativa de la red. El método *createRandomBN()* será el encargado de crear cada variable de la red y añadirla para crear el objeto *ProNet* que representa la red. el método *getTablePotencial()* calculará la tabla de probabilidad condicionada asociada a cada variable de la red teniendo en cuenta cada nodo y sus antecesores según hemos explicado al principio de este mismo apartado.

Capítulo 8

Pruebas y resultados

8.1. Test de Unidad

En este apartado vamos a comentar cada uno de los test que se han definido para probar la librería implementada y comprobar que se han alcanzado los objetivos marcados cumpliendo con los requisitos establecidos en la concepción de este proyecto. La librería implementada incorpora un paquete donde se agrupan las clases de test que nos van a permitir ejecutar las pruebas unitarias sobre cada uno de los tipos de pregunta que se generan en la librería. A continuación vamos a comentar el contenido de dicho paquete de pruebas unitarias *es.uned.dia.pfc.lgap.mgp.test*.

8.1.1. Paquete *test*

En los apartados siguientes vamos a comentar cada una de las clases del *framewok* de JUnit que nos van a permitir realizar las pruebas unitarias o test de unidad de la librería, de cada tipo de pregunta que se puede plantear dentro del repertorio ofrecido por la librería. Cada clase de test hereda de la clase *TestCase* del *framework* de JUnit y tendremos un test para cada tipo de pregunta del repertorio ofrecido por la librería. El test consistirá en crear una instancia concreta del tipo de pregunta y validar los métodos de su interface con el sistema *Siette*.

Los test o pruebas se han organizado también por bloques temáticos agrupando cada test en un bloque en función del tipo de pregunta que se plantea. Vamos a ver a continuación cada uno de estos test en el ámbito de su bloque temático.

8.1.2. Pruebas bloque temático 1

En el diagrama 8.1 se representan las clases de test que se han definido para cada tipo de pregunta que se plantea sobre el bloque temático 1 que tiene que ver con las relaciones de independencia probabilística y de independencia probabilística condicional entre las variables aleatorias de un sistema a partir de una distribución de probabilidad dada.

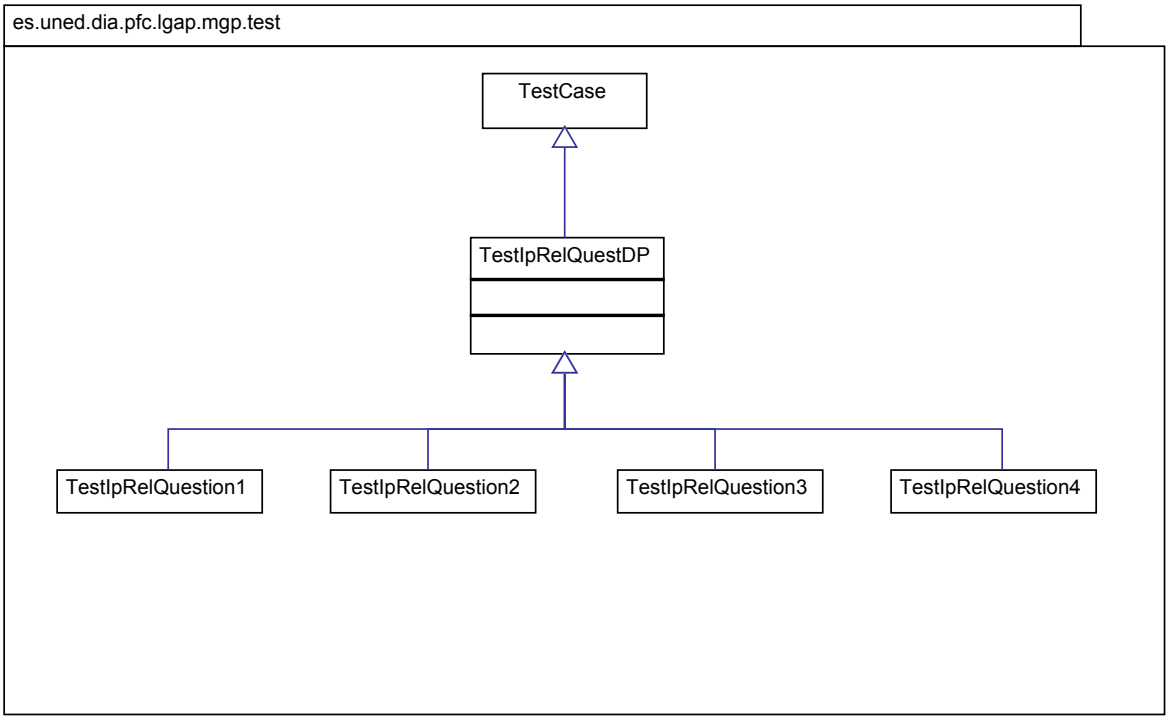


Figura 8.1: Diagrama de clases de test sobre Distribuciones de Probabilidad y relaciones de independencia.

8.1.2.1. Caso de prueba 1

En este caso de prueba se testea la creación de preguntas de tipo 1. Para ello se crea una instancia de la clase *IpRelQuestion1*. La prueba consiste en la generación de un fichero PDF con el contenido de la pregunta: enunciado y posibles respuestas junto con el refuerzo correspondiente a las respuestas correctas e incorrectas. Mediante la creación de un objeto del tipo *ExamenTest* invocamos al constructor que recibe la instancia de la pregunta generada y se genera el correspondiente archivo PDF con el resultado de imprimir la pregunta. El resto de métodos del test permiten probar los métodos restantes en especial los que sirven de interfaz con la librería *Siette*.

8.1.2.2. Caso de prueba 2

Análogamente al caso del apartado anterior, creamos una instancia de la clase *IpRelQuestion2*. Mediante la invocación al método *genPDFLatexFile()* de la clase *ExamenTest* generamos un fichero en formato PDF con el resultado de imprimir la pregunta de tipo 2 generada en la instanciación.

8.1.2.3. Caso de prueba 3

Igual que en los apartados anteriores se define el caso de prueba con la clase *IpRelQuestion3.*

8.1.2.4. Caso de prueba 4

Igual que en los apartados anteriores se define el caso de prueba con la clase *IpRelQuestion4.*

8.1.3. Pruebas bloque temático 2

En este apartado consideramos cuatro tipos de pruebas unitarias. Las dos primeras correspondientes a los dos primeros apartados, tratan de validar la generación de grafos y particularmente su representación gráfica tanto de grafos dirigidos como de grafos no dirigidos. En los dos apartados siguientes se testean los tipos de preguntas correspondientes al bloque temático 2 que plantea preguntas que tienen que ver con relaciones de independencia probabilística y de independencia probabilística condicional entre variables o nodos de un grafo acíclico dado (dirigido o no dirigido).

En la figura 8.2 se representa el diagrama de clases correspondiente a los casos de prueba que se comentarán en los apartados siguientes:

8.1.3.1. Caso de prueba 5.

En este apartado nos vamos a centrar en las pruebas de generación y representación gráfica de Grafos No dirigidos Acíclicos (GNA). La prueba consiste en generar un grafo no dirigido acíclico, se crea una instancia de la clase *GrafoGNA* y se obtiene su representación gráfica mediante su generación de código *SVG*. Al mismo tiempo se calculan los caminos activos entre dos nodos dados. El cálculo de los caminos activos entre dos nodos es la base para las preguntas que se van a plantear posteriormente sobre las relaciones de independencia probabilística y de independencia probabilística condicional entre dos nodos respecto a un tercero.

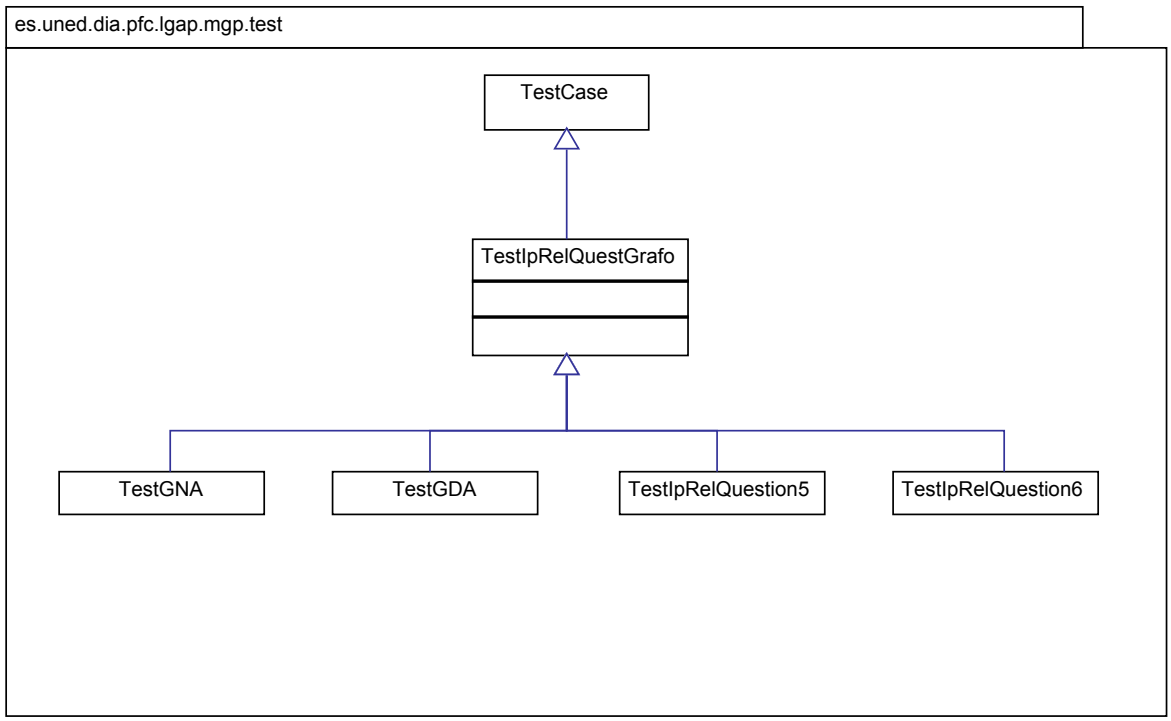


Figura 8.2: Diagrama de clases de test sobre Grafos y relaciones de independencia.

8.1.3.2. Caso de prueba 6

En este caso vamos a testear la generación y representación gráfica de los Grafos Dirigidos Acíclicos (GDA). Análogamente a lo explicado en el apartado anterior, en este caso vamos a testear los caminos activos entre dos nodos de un grafo para verificar si se cumplen o no las propiedades de independencia probabilística y de independencia probabilística condicional entre los nodos de un grafo que representan variables aleatorias. Esto nos va a permitir dar con la respuesta adecuada a las preguntas que se plantearán posteriormente sobre el bloque temático 2.

8.1.3.3. Caso de prueba 7

En este apartado testearemos los tipos de pregunta de tipo 5 (ver 7.3.2.4) que se definen sobre grafos no dirigidos acíclicos. Una vez se ha testeado la generación y representación gráfica junto con el cálculo de los caminos activos ente dos nodos, pasamos a crear distintas instancias de la clase *ipRelQuiestion5*. Al igual que en el resto de las pruebas sobre los distintos tipos de pregunta, se generará un fichero en formato PDF con el contenido de la pregunta generada.

8.1.3.4. Caso de prueba 8

En este último test del bloque temático 2 se testará la clase *ipRelQuestion6* que plantea preguntas sobre relaciones de independencia probabilística y de independencia probabilística condicional entre dos nodos de un grafo mediante la generación del correspondiente archivo PDF con el contenido de la pregunta.

8.1.4. Pruebas bloque temático 3

El tercer bloque temático incluye igualmente una serie de clases 8.3 que nos van a permitir realizar las pruebas unitarias sobre los distintos tipos de pregunta que se plantean en esta parte.

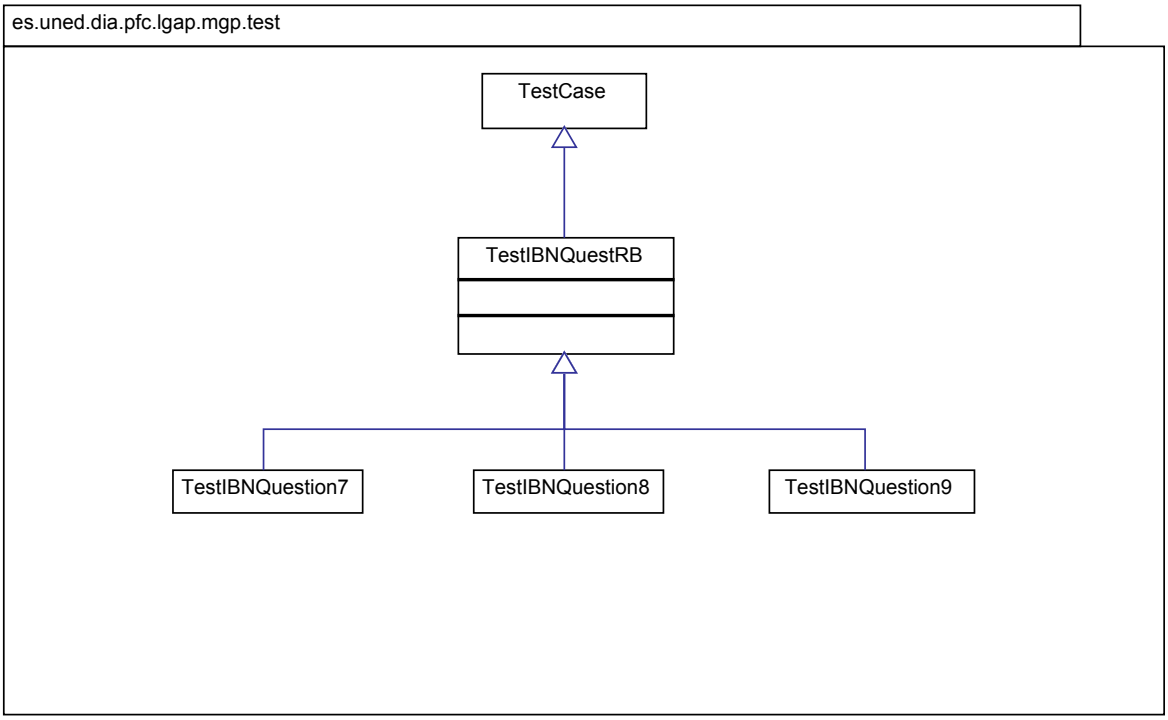


Figura 8.3: Diagrama de clases de test sobre inferencia en redes bayesianas.

8.1.4.1. Caso de prueba 9

En este caso de prueba se testea la creación de preguntas de tipo 7 que tiene que ver con el estudio de la red bayesiana *Disease/Test*. Para ello se crea una instancia de la clase *IBNQuestion7*. La prueba consiste en la generación de un fichero PDF con el contenido de la pregunta: enunciado y posibles respuestas junto con el refuerzo correspondiente a las

respuesta correcta. Mediante la creación de un objeto del tipo *ExamenTest* invocamos al constructor que recibe la instancia de la pregunta generada y se genera el correspondiente archivo PDF con el resultado de imprimir la pregunta. El resto de métodos del test permiten probar los métodos restantes en especial los que sirven de interfaz con la librería *Siette*.

8.1.4.2. Caso de prueba 10

En este caso de prueba se testea la creación de preguntas de tipo 8 que tiene que ver con el estudio de las redes bayesianas ofrecidas por la factoría de *OpenMarkov*. Para ello se crea una instancia de la clase *IBNQuestion8*. La prueba consiste en la generación de un fichero PDF con el contenido de la pregunta: enunciado y posibles respuestas junto con el refuerzo correspondiente a las respuesta correcta. Mediante la creación de un objeto del tipo *ExamenTest* invocamos al constructor que recibe la instancia de la pregunta generada y se genera el correspondiente archivo PDF con el resultado de imprimir la pregunta. El resto de métodos del test permiten probar los métodos restantes en especial los que sirven de interfaz con la librería Siette.

8.1.4.3. Caso de prueba 11

En este caso de prueba se testea la creación de preguntas de tipo 9 que tiene que ver con el estudio de la redes bayesianas de generación aleatoria. Para ello se crea una instancia de la clase *IBNQuestion9*. La prueba consiste en la generación de un fichero PDF con el contenido de la pregunta: enunciado y posibles respuestas junto con el refuerzo correspondiente a las respuesta correcta. Mediante la creación de un objeto del tipo *ExamenTest* invocamos al constructor que recibe la instancia de la pregunta generada y se genera el correspondiente archivo PDF con el resultado de imprimir la pregunta. El resto de métodos del test permiten probar los métodos restantes en especial los que sirven de interfaz con la librería *Siette*.

8.2. Test de Eficiencia

Uno de los objetivos del presente proyecto de fin de carrera es estudiar la eficiencia de la librería desarrollada de manera que pueda ser utilizada en un entorno de tiempo real donde los alumnos necesitan una rápida respuesta en la generación de cada pregunta cuando están realizando un test de autoevaluación. La eficiencia de la librería tendrá que ver asimismo con la eficiencia del sistema *Siette* en el cuál se integra, siendo éste un sistema

que se ejecuta en un servidor para una aplicación web y teniendo en cuenta que los alumnos acceden a dicho sistema a través de su navegador y una determinada conexión a Internet que determinará un ancho de banda, factores estos que influirán significativamente en la eficiencia global del sistema a la hora de realizar los ejercicios o test de autoevaluación.

Teniendo en cuenta los factores considerados en el párrafo anterior, para facilitar la recolección de los datos se han realizado las ejecuciones desde el entorno de desarrollo Eclipse utilizado en el proyecto para Java, tomando los tiempos que tardan en ejecutarse cada uno de los test unitarios que se han comentado en la sección anterior 8.1. El paquete *es.uned.ia.pfc.lgap.mgp.eficiencia* reúne una serie de test que nos permiten medir la eficiencia de cada uno de los tipos de preguntas y de las instancias de las mismas que se definen en la asignatura *Siette*.

En esta sección vamos a realizar un estudio de la eficiencia de la librería desarrollada atendiendo a los tiempos que tarda en la generación de cada tipo de pregunta perteneciente a cada uno de los bloques temáticos. Se realizará una medición del tiempo peor, el mejor tiempo, el promedio en la generación de cada tipo de pregunta, así como de la varianza. Con los datos obtenidos, se calculará la desviación estándar como medida de dispersión. En los apartados siguientes vamos a realizar el estudio por partes atendiendo a la división en bloques temáticos que se ha realizado en este proyecto y de los distintos tipos de preguntas que se han definido para cada bloque temático (ver A.1).

Como se ha comentado, en los tiempos influyen varios factores como las características de la máquina en la que se realizan las ejecuciones, o de la carga y características del servidor cuando se ejecuta en *Siette*, del navegador del cliente, del tráfico de datos en la red, etc. En todas las mediciones realizadas se ha despreciado el valor del tiempo obtenido para las primeras ejecuciones de cada pregunta debido a que son significativamente más altas que el resto de la muestra. Esto es así debido a la preparación que requiere la máquina virtual Java en la primera ejecución cuando se cargan las clases, se crean los objetos, etc.

8.2.1. Eficiencia en bloque temático 1

En este primer apartado vamos a realizar el estudio de la eficiencia correspondiente al primer bloque de preguntas. Este bloque está compuesto por cuatro tipos de preguntas que se corresponden con los test unitarios realizados en el apartado 8.1.2. Para cada tipo de pregunta vamos a considerar tres instancias que tienen una correspondencia directa con las preguntas definidas la asignatura del sistema *Siette*. A continuación vamos a mostrar los resultados correspondientes a la generación de cada uno de los tipos de pregunta comentados en cuanto al tiempo que tardan en su generación. Cada tabla muestra un

Tabla 8.1: Resultados de la generación de preguntas de tipo 1 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
1_1	0,29	13	0,76	1,41	1,19	764,9
1_2	0,32	3,05	0,46	0,08	0,29	458,19
1_3	0,87	30,64	2,49	2,29	1,51	2.491,46

tipo de pregunta determinado.

8.2.1.1. Preguntas de tipo 1

En la tabla 8.1 se muestran los resultados obtenidos para cada instancia del primer tipo de pregunta. Los resultados se obtienen tras la ejecución del test JUnit *TestIpRelEficiencia1* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 1_1, 1_2 y 1_3), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 3.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.4 se representa el gráfico correspondiente a los datos de la tabla 8.1.

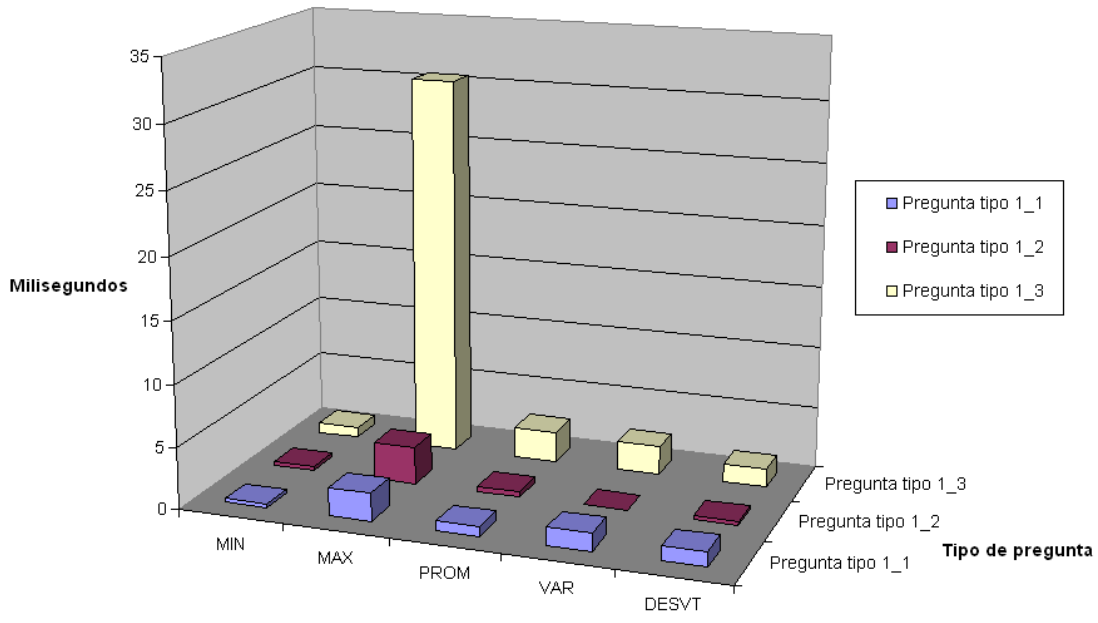


Figura 8.4: Eficiencia Tipo de pregunta 1.

Tabla 8.2: Resultados de la generación de preguntas de tipo 2 (tiempo en milisegundos)

Instancia	**Mínimo**	**Máximo**	**Promedio**	**Varianza**	**Desviación típica**	**Total**
2_1	0,16	10,59	0,56	0,66	0,81	588,82
2_2	0,27	6,6	0,43	0,15	0,39	427,43
2_3	0,01	8,56	0,93	0,27	0,52	933,33

A la vista de los resultados podemos observar como la generación de las preguntas tipo 1_1 y 1_2 son similares debido a que sólo se diferencian en el número de alternativas a presentar, algo que no influye en el tiempo de generación. Por su parte las preguntas de tipo 1_3 experimentan un considerable incremento del tiempo máximo, manteniéndose estables los demás valores debido a la incorporación de una nueva variable aleatoria en el sistema haciendo un total de cuatro (en las preguntas de tipo 1_1 y 1_2 se trabaja con tres variables aleatorias). La distribución de probabilidad que se define entre las variables aleatorias del sistema aumenta el número de combinaciones entre los distintos valores que toman las variables generándose por tanto una mayor complejidad en la generación de dicha distribución de probabilidad y consecuentemente un incremento en el tiempo de generación de este tipo de preguntas.

8.2.1.2. Preguntas de tipo 2

En la tabla 8.2 se muestran los resultados obtenidos para cada instancia del segundo tipo de pregunta. Los resultados se obtienen tras la ejecución del test JUnit *TestIpRelEficiencia2* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 2_1, 2_2 y 2_3), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 3.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.5 se representa el gráfico correspondiente a los datos de la tabla 8.2.

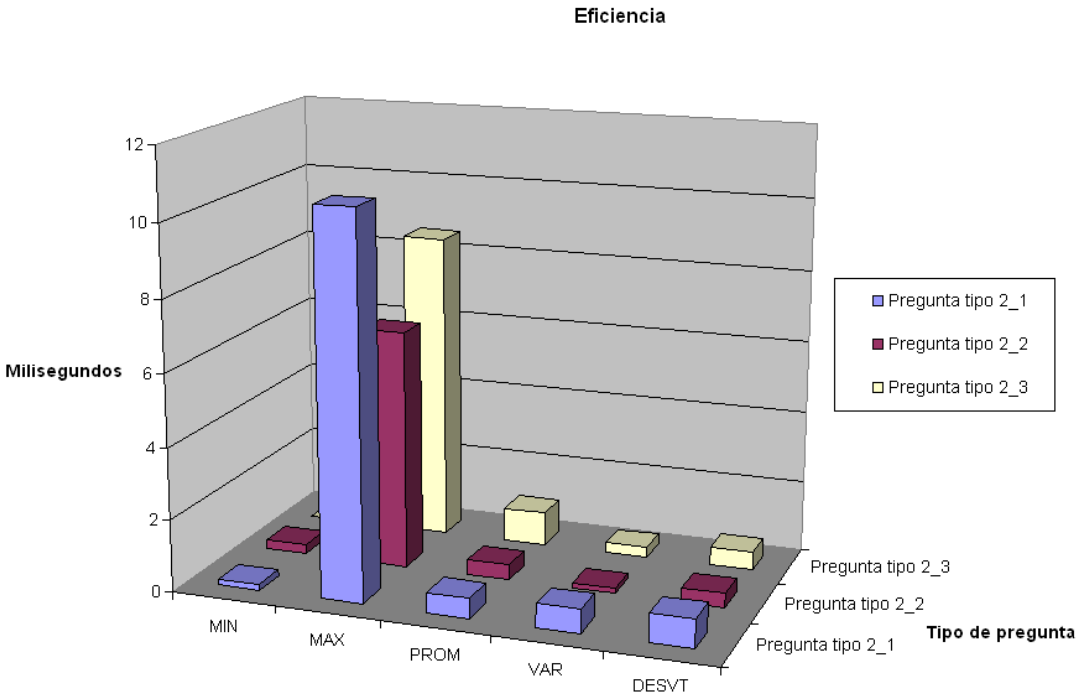


Figura 8.5: Eficiencia Tipo de pregunta 2.

La interpretación de los resultados en este caso pasa por observar como la generación de los distintos tipos de preguntas son similares, ya que en todas ellas se tienen que generar cuatro distribuciones de probabilidad sobre las variables aleatorias del sistema (ver 6.2.1.2).

8.2.1.3. Preguntas de tipo 3

En la tabla 8.3 se muestran los resultados obtenidos para cada instancia del tercer tipo de pregunta. Los resultados se obtienen tras la ejecución del test JUnit *TestIpRelEficiencia3* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 3_1, 3_2 y 3_3), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 3.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.6 se representa el gráfico correspondiente a los datos de la tabla 8.3.

Tabla 8.3: Resultados de la generación de preguntas de tipo 3 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
3_1	0,23	8,84	0,42	0,61	0,78	423,97
3_2	0,19	5,1	0,26	0,08	0,28	262,42
3_3	0,03	1,87	2,11	0,02	0,13	107,49

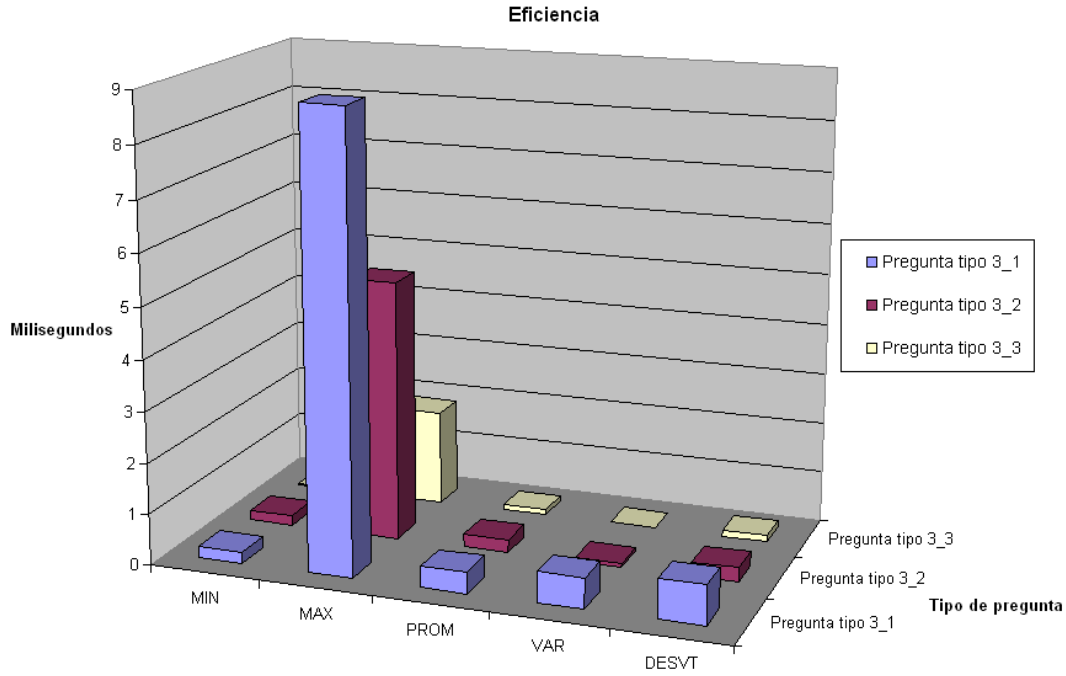


Figura 8.6: Eficiencia Tipo de pregunta 3.

En el gráfico de la figura 8.6 podemos observar que las preguntas de tipo 3_3 disminuyen el tiempo de ejecución. Esto es debido a que el sistema generado se compone de dos variables aleatorias en lugar de las tres que se generan para las preguntas de tipo 3_1 y 3_2. Concluimos por tanto que influye en el tiempo de ejecución el número de variables aleatorias de las que se compone el sistema considerado en cada tipo de pregunta.

8.2.1.4. Preguntas de tipo 4

En la tabla 8.4 se muestran los resultados obtenidos para cada instancia del cuarto tipo de pregunta. Los resultados se obtienen tras la ejecución del test JUnit *TestIpRelEficiencia4* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indica-

Tabla 8.4: Resultados de la generación de preguntas de tipo 4 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
4_1	0,35	13,84	0,74	1,47	1,21	736,7
4_2	0,09	9,74	0,44	0,19	0,44	439,84
4_3	0,15	3,81	0,54	0,16	0,40	540,04

dos en la tabla (Instancias 4_1, 4_2 y 4_3), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 3.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.7 se representa el gráfico correspondiente a los datos de la tabla 8.4.

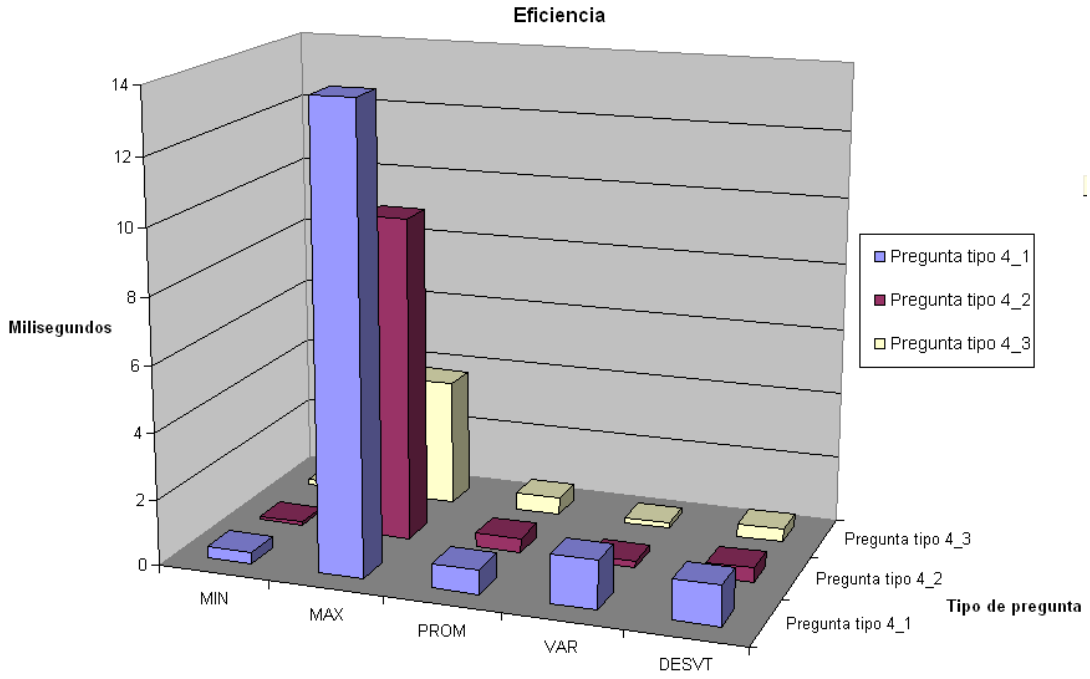


Figura 8.7: Eficiencia Tipo de pregunta 4.

En este caso observamos un tiempo de ejecución algo menor de las preguntas de tipo 4_3 respecto a las otras dos cuando se utilizan variables aleatorias combinadas con variables binarias.

Tabla 8.5: Resultados de la generación de preguntas de tipo 5 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
5_1	0,08	9,41	0,21	0,43	0,66	211,19
5_2	0,09	3,02	0,15	0,04	0,19	152,47
5_22	0,82	16,95	1,93	0,8	0,9	1.933,06
5_3	0,05	3,53	0,10	0,04	0,2	95,04
5_4	0,07	1,71	0,14	0,02	0,15	144,29
5_44	0,07	3,36	0,11	0,02	0,16	113,63
5_5	0,03	2,32	0,08	0,02	0,13	79,11
5_6	0,22	2,67	0,12	0,03	0,18	115,68
5_66	0,86	28,82	1,04	1,05	1,02	1041,66

8.2.2. Eficiencia en bloque temático 2

En este segundo apartado vamos a realizar el estudio de la eficiencia correspondiente al segundo de los bloques de preguntas considerados en este proyecto. Dicho bloque está compuesto por dos nuevos tipos de preguntas que se corresponden con los test unitarios realizados en el apartado 8.1.3. Para cada tipo de pregunta vamos a considerar nueve instancias que tienen una correspondencia directa con las preguntas definidas en el sistema *Siette*. A continuación vamos a mostrar los resultados correspondientes a la generación de cada uno de los tipos de pregunta comentados en cuanto al tiempo que tardan en su generación. Cada tabla muestra un tipo de pregunta determinado.

8.2.2.1. Preguntas de tipo 5

En la tabla 8.5 se muestran los resultados obtenidos para cada instancia del tipo de pregunta 5. Los resultados se obtienen tras la ejecución del test JUnit *TestIpRelEficiencia5* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 5_1, 5_2, 5_22, 5_3, 5_4, 5_44, 5_5, 5_6 y 5_66), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 9.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta que se presentan en este apartado del estudio.

En la figura 8.8 se representa el gráfico correspondiente a los datos de la tabla 8.5.

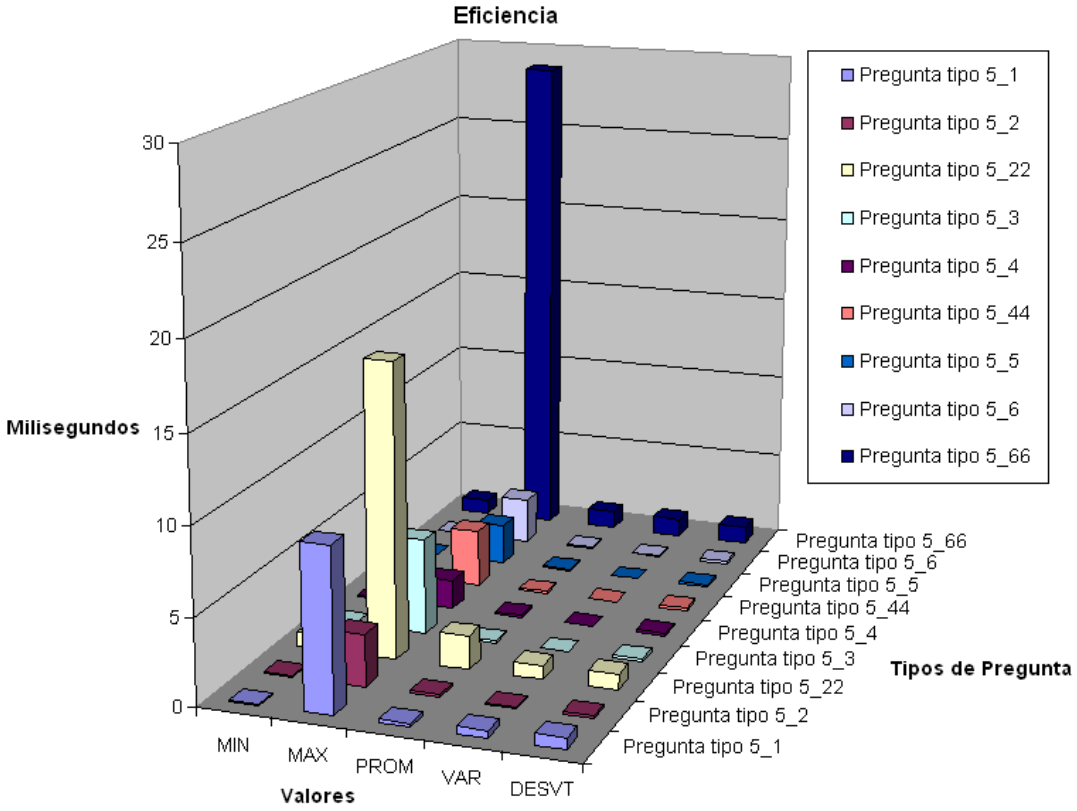


Figura 8.8: Eficiencia Tipo de pregunta 5.

Observando el gráfico de la figura 8.8 podemos observar como la generación de las preguntas que eligen aleatoriamente los parámetros para el número de nodos, enlaces y la inclusión de las relaciones de independencia probabilística condicional, preguntas de tipo 5_22, 5_44 y 5_66 son las que consumen un mayor tiempo en su generación, alcanzándose los valores más altos en los máximos correspondientes.

8.2.2.2. Preguntas de tipo 6

En la tabla 8.6 se muestran los resultados obtenidos para cada instancia del tipo de pregunta 6. Los resultados se obtienen tras la ejecución del test JUnit *TestIpRelEficiencia6* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 6_1, 6_2, 6_22, 6_3, 6_4, 6_44, 6_5, 6_6 y 6_66), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 9.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta que se presentan

Tabla 8.6: Resultados de la generación de preguntas de tipo 6 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
6_1	0,04	9,41	0,29	0,55	0,74	289,81
6_2	0,11	15,85	0,22	0,46	0,68	221,06
6_22	0,06	1,16	0,09	0,10	0,31	91,79
6_3	0,09	24,94	0,15	0,63	0,8	151,23
6_4	0,08	15,54	0,10	0,28	0,53	203,38
6_44	0,14	19,3	0,21	0,49	0,7	210,74
6_5	0,17	18,58	0,12	0,46	0,68	121,35
6_6	0,06	27,59	0,15	0,78	0,88	150,3
6_66	0,09	17,96	0,14	0,35	0,59	138,02

en este apartado del estudio.

En la figura 8.9 se representa el gráfico correspondiente a los datos de la tabla 8.6.

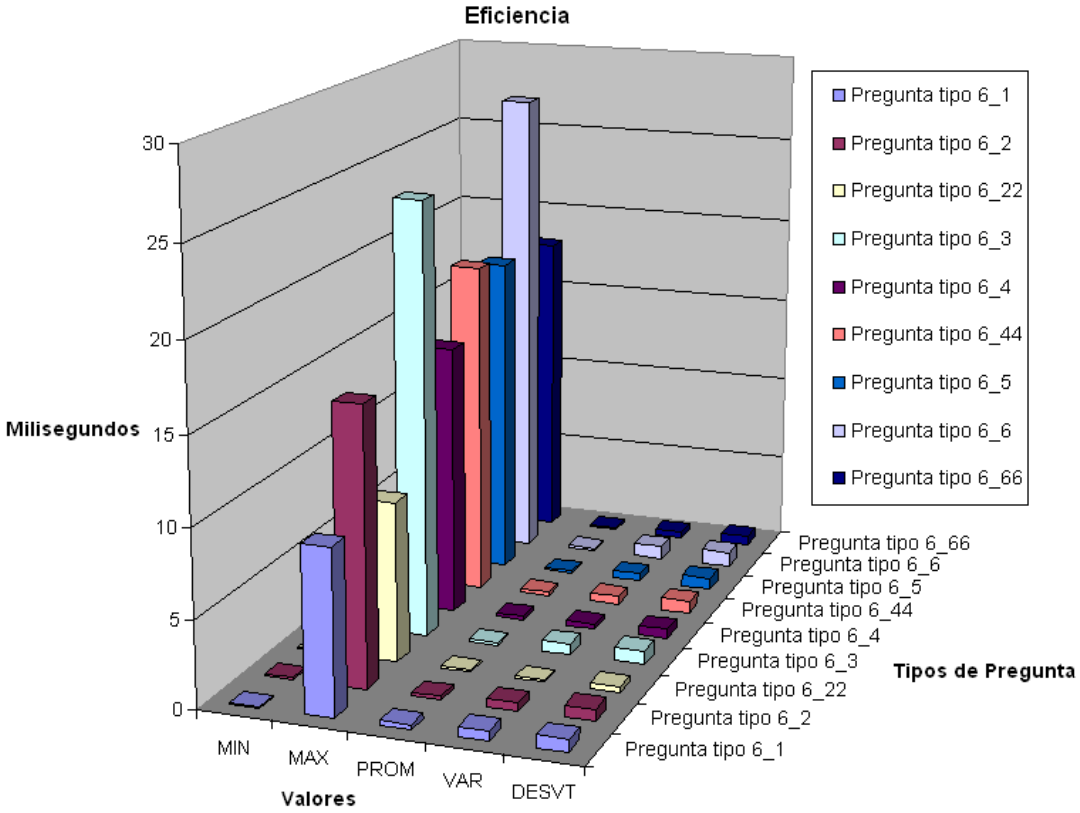


Figura 8.9: Eficiencia Tipo de pregunta 6.

Teniendo en cuenta el gráfico de la figura 8.9 de nuevo podemos observar como las preguntas que trabajan con grafos con un mayor número de nodos son los que consumen

Tabla 8.7: Resultados de la generación de preguntas de tipo 7 (tiempo en milisegundos)

Instancia	**Mínimo**	**Máximo**	**Promedio**	**Varianza**	**Desviación típica**	**Total**
7_1	0,23	15,26	0,27	0,57	0,75	266,31
7_2	0,19	30,13	0,12	0,94	0,97	124,43

mayor tiempo. Esto es debido a la exigencia de que los enlaces entre los nodos han de ser elegidos de manera que no se formen ciclos dentro del grafo y que este sea conexo, es decir todos los nodos estén conectados.

8.2.3. Eficiencia en bloque temático 3

En esta última parte nos centramos en las preguntas de inferencia en redes bayesianas que hacen uso del API de *OpenMarkov*. En este bloque encontramos los tres últimos tipos de preguntas que se corresponden con los test unitarios realizados en el apartado 8.1.4. Para cada tipo de pregunta vamos a considerar una serie de instancias que tienen una correspondencia directa con las preguntas definidas en el sistema *Siette*. En las tablas y gráficos siguientes se muestran los resultados correspondientes a la generación de cada uno de los tipos de pregunta comentados en cuanto al tiempo que tardan en su generación. Cada tabla muestra un tipo de pregunta determinado.

8.2.3.1. Preguntas de tipo 7

En la tabla 8.7 se muestran los resultados obtenidos para cada instancia del tipo de pregunta 7. Los resultados se obtienen tras la ejecución del test JUnit *TestIBNEficiencia7* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 7_1 y 7_2), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 2.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.10 se representa el gráfico correspondiente a los datos de la tabla 8.7.

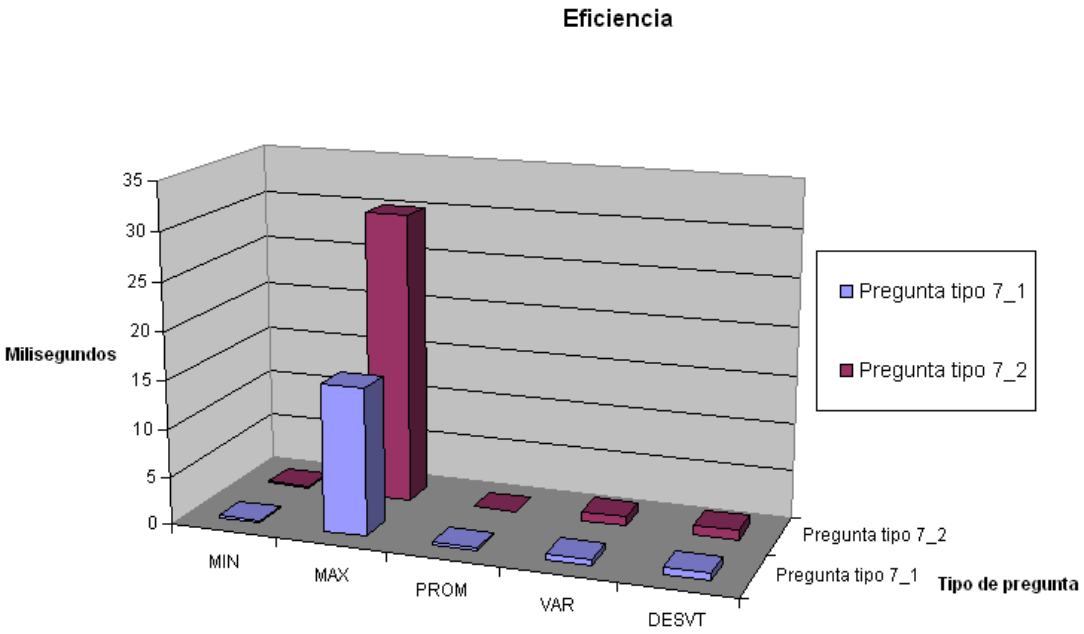


Figura 8.10: Eficiencia Tipo de pregunta 7.

El simple hecho de aumentar la precisión en los cálculos, pasar de utilizar dos decimales a utilizar cuatro, repercute en el tiempo de generación de una pregunta de este tipo. Observamos como la pregunta de tipo 7_2 tarda más en generarse que la 7_1.

8.2.3.2. Preguntas de tipo 8

En la tabla 8.8 se muestran los resultados obtenidos para cada instancia del tipo de pregunta 8. Los resultados se obtienen tras la ejecución del test JUnit *TestIBNEficiencia8* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados en la tabla (Instancias 8_1, 8_2, 8_3 y 8_4), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 4.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.11 se representa el gráfico correspondiente a los datos de la tabla 8.8.

Tabla 8.8: Resultados de la generación de preguntas de tipo 8 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
8_1	0,23	22,97	1,37	3,88	1,97	1.370,33
8_2	0,19	11,81	0,74	0,31	0,56	743,35
8_3	60,83	231,91	67,50	210,29	14,51	67.503,23
8_4	0,03	5,64	0,7	0,23	0,48	699,76

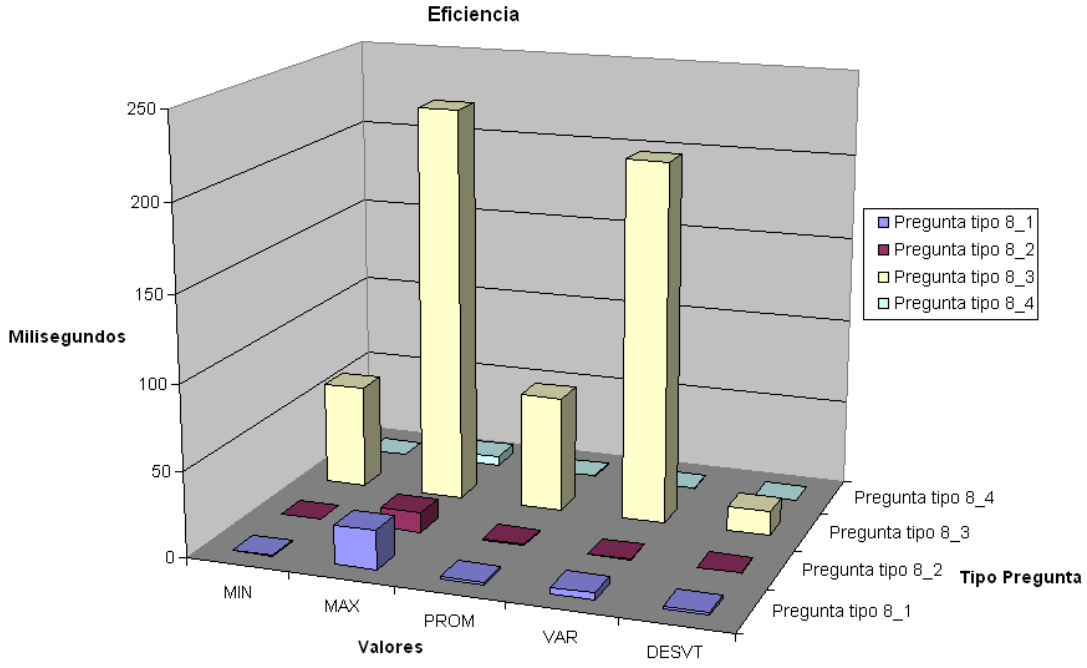


Figura 8.11: Eficiencia Tipo de pregunta 8.

Observamos que la pregunta 8_3 para la generación de la red bayesiana de la factoría *OpenMarkov* Asia, como era de esperar, es la que más tiempo consume en la generación debido a la complejidad de la misma en comparación con las otras dos redes *bN_ABC* y *bN_XYZ*. Observamos también que aumenta la varianza al hacerse más significativa la diferencia entre el valor mínimo y el máximo en el tiempo de ejecución.

8.2.3.3. Preguntas de tipo 9

En la tabla 8.9 se muestran los resultados obtenidos para cada instancia del tipo de pregunta 9. Los resultados se obtienen tras la ejecución del test JUnit *TestIBNEficiencia9* del paquete *es.uned.dia.pfc.lgap.mgp.eficiencia*. Dicho test realiza 1.000 ejecuciones del mismo dando lugar a la generación de 1.000 preguntas de cada uno de los tipos indicados

Tabla 8.9: Resultados de la generación de preguntas de tipo 9 (tiempo en milisegundos)

Instancia	Mínimo	Máximo	Promedio	Varianza	Desviación típica	Total
9_1	0,21	15,42	0,93	1,66	1,29	929,36
9_2	0,18	8,56	1,87	0,77	0,88	1.870,06
9_3	0,05	8,74	2,3	0,87	0,93	2.300,24

en la tabla (Instancias 9_1, 9_2 y 9_3), se muestra el mínimo tiempo en la generación de cada pregunta, el mayor, el promedio, la varianza y la desviación típica o estándar de los valores. En la columna Total se muestra el tiempo registrado en la ejecución total del test de eficiencia. En este caso el resultado de las 3.000 ejecuciones, 1.000 para cada una de las instancias o tipos de pregunta.

En la figura 8.12 se representa el gráfico correspondiente a los datos de la tabla 8.9.

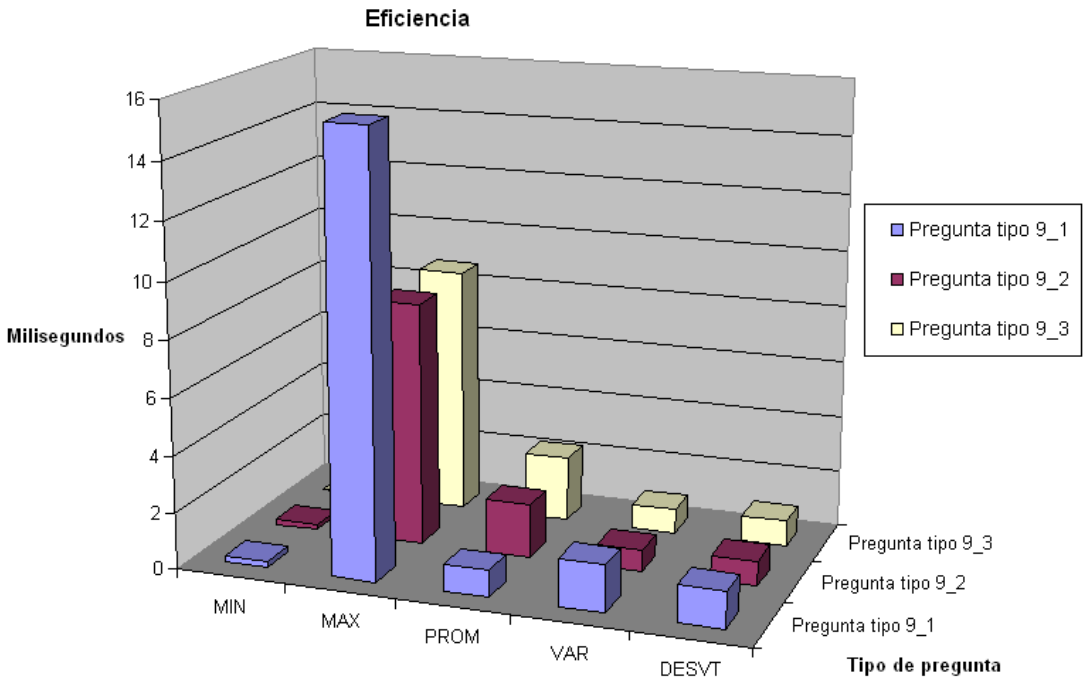


Figura 8.12: Eficiencia Tipo de pregunta 9.

En este caso observamos unos resultados más equilibrados entre los distintos tipos de preguntas. El mayor o menor tiempo en la generación de cada pregunta estará directamente relacionado con el número de nodos y enlaces de los que se compone la red bayesiana generada.

Capítulo 9

Conclusiones y líneas de trabajo futuras

9.1. Conclusiones

Una vez finalizadas la fase de implementación y pruebas de la librería *LGAP*, se comprueba que se han cumplido los objetivos marcados. Se ha constatado que el sistema desarrollado es fiel a los requisitos marcados, superándolos incluso en muchos aspectos.

La arquitectura del sistema aporta la característica de ser fácilmente escalable, resulta relativamente sencillo introducir nuevos tipos de preguntas sobre el mismo dominio o bloque temático o sobre otros nuevos. Se han empleado las tecnologías, estándares y metodologías marcadas como objetivo. A partir de la librería software desarrollada, el modo en que esta ha sido diseñada, deja abierta la posibilidad de realizar trabajos de ampliación futuros que permitan enriquecerla con nuevos tipos de ejercicios o preguntas de generación automática. Para el análisis y diseño se han utilizado técnicas de arquitectura basadas en UML y patrones de diseño y se ha configurado un marco de referencia o *framework* específico de desarrollo para incorporar nuevos tipos de pregunta que sean integrables dentro del sistema *Siette*. Se ha implementado un sistema de trazas de ejecución. Se ha definido una jerarquía propia de excepciones para el control de excepciones de la aplicación. Se ha documentado el sistema utilizando *Javadoc*.

Para el desarrollo del sistema se han utilizado entornos de desarrollo, metodologías, herramientas y lenguajes de código abierto, y de gran difusión entre la comunidad de desarrolladores y específicamente de estudiantes de ingeniería en informática siguiendo el compromiso del equipo docente del departamento de Inteligencia Artificial de la UNED donde se enmarca el presente proyecto. El desarrollo se ha realizado en Java utilizando

Eclipse como *IDE* con diversos complementos como *Tomcat* para el despliegue de la librería y la realización de pruebas unitarias con el *framework JUnit*. Objetos de diversas clases, entre los que cabe destacar paquetes de sistema y manejo de grafos, que constituyen la base para la definición de las redes Bayesianas. Se ha implementado un sistema de trazas de ejecución, utilizando la *API Java Logging*. Se ha definido una jerarquía propia de excepciones para el control de excepciones de la aplicación. Se ha documentado el sistema utilizando Javadoc.

9.2. Trabajo futuro. Ampliaciones de la librería

La librería desarrollada es perfectamente integrable en el sistema *Siette* a través de la interfaz definida en el paquete que forma parte de la librería (ver 7.1.3). Un profesor de la asignatura Métodos Gráficos Probabilistas introducirá, a través de la gestión de archivos de *Siette*, la librería *LGAP* invocará a los métodos ofrecidos por la interfaz *Siette* para dar soporte a la definición de cada una de los distintos tipos de preguntas implementadas en la librería. Variando la parametrización de cada objeto del tipo de pregunta se generarán distintas instancias u objetos de tipo pregunta que darán lugar a las distintas preguntas *Siette* que posteriormente formarán parte de los test de evaluación que serán resueltos por los alumnos y usuarios del sistema (ver apéndice A para las distintas opciones de parametrización y los tipos de preguntas disponibles en la librería).

Asimismo el desarrollo del programa cliente que hace uso de la librería constituye una herramienta muy útil para un profesor de la asignatura que desee realizar exámenes escritos de evaluación a sus alumnos con tan sólo pulsar un botón. El programa genera el correspondiente examen en formato PDF para ser resueltos pos los alumnos. El profesor podrá realizar la corrección de los mismos generando el PDF con las soluciones a cada una de las preguntas planteadas.

Al hilo del diseño realizado y del camino que se ha marcado, la ampliación pasaría por definir nuevas clases en el sistema que permitan considerar nuevos conceptos distintos a los considerados, definir nuevos bloques temáticos dentro de los cuáles se enmarcaría la definición de nuevos tipos de preguntas que ampliarían el repertorio ofrecido por la librería o servirían para la generación de nuevas librerías similar a la desarrollada para la definición de nuevas asignaturas que se integrarían en el sistema *Siette*.

Apéndice A

Tipos de preguntas en *Siette*

En este apéndice vamos a especificar los distintos tipos de preguntas que exporta la librería desarrollada para su integración en el sistema *Siette*. La asignatura definida será **Modelos Gráficos Probabilistas**. La librería *LgapMgp* aportará nueve tipos distintos de preguntas para esta asignatura. Cada tipo de pregunta ofrece una determinada parametrización que nos permitirá definir varias preguntas finales de la asignatura indicada en el sistema. En la tabla A.1 comentamos los distintos tipos de preguntas y su parametrización para la definición de la asignatura en el sistema *Siette*.

Tabla A.1: Tipos de pregunta ofrecidos por la librería *LgapMgp* para *Siette*.

Pregunta	**Clase**	**Parametrización**
Pregunta tipo 1	IpRelQuestion1	precisión, nº variables, tipo, nº alternativas, nº correctas
Pregunta tipo 2	IpRelQuestion2	precisión, nº variables, tipo, nº alternativas, nº correctas
Pregunta tipo 3	IpRelQuestion3	precisión, nº variables, tipo, nºalternativas, nº correctas
Pregunta tipo 4	IpRelQuestion4	precisión, nº variables, tipo, nº incógnitas
Pregunta tipo 5	IpRelQuestion5	nombre, nº nodos, nº enlaces, condicional, resp.abierta, nº correctas, nº alternativas
Pregunta tipo 6	IpRelQuestion6	nombre, nº nodos, nº enlaces, condicional, resp.abierta, nº correctas, nº alternativas
Pregunta tipo 7	IBNQuestion7	nombre, precisión
Pregunta tipo 8	IBNQuestion8	nombre, precisión, nº alternativas, tipo de red
Pregunta tipo 9	IBNQuestion9	nombre, precisión, nº nodos, nº enlaces,nº alternativas

Como se ha comentado a lo largo de toda la memoria del presente proyecto, el desarrollo de la librería se ha estructurado en tres bloques temáticos. De la misma manera se definirán en *Siette* tres temas dentro de la asignatura, cada uno de ellos se corresponderá con un bloque temático de los aportados por la librería. Los temas definidos son los siguientes:

1. Conceptos básicos de dependencia e independencia probabilística entre variables

aleatorias de un sistema.

2. Dependencia e independencia probabilística en grafos.

3. Inferencia en Redes Bayesianas.

En los siguientes apartados comentamos cada uno de estos tres bloques temáticos.

A.1. Primer bloque temático

Las preguntas de tipo 1, 2, 3 y 4 pertenecen al primer bloque temático. En este tipo de preguntas se plantean relaciones de independencia e independencia probabilística condicional entre variables aleatorias de un sistema formado por un número determinado de variables aleatorias discretas sobre una distribución de probabilidad dada. Cada tipo de pregunta presenta una serie de parámetros que nos permiten obtener variaciones sobre la pregunta obteniendo distintas instancias de la pregunta en función de los parámetros elegidos. Los parámetros para este tipo de preguntas son los siguientes:

- **precisión**: número de decimales que se van a utilizar en los cálculos y en la representación de los distintos valores numéricos asociados a los valores numéricos de las probabilidades.

- **nº variables**: número de variables que forman parte del sistema planteado en el enunciado.

- **tipo**: tipo de variables que forman parte del sistema. Estas pueden ser variables binarias (sólo pueden tomar dos posibles valores), variables genéricas o una combinación de ambas. [1]

- **nº alternativas**: número de respuestas o alternativas que se plantean como posibles respuestas a la pregunta planteada y de las cuales el alumno deberá elegir la respuesta o respuestas correctas en función del tipo de pregunta: pregunta de respuesta simple o pregunta de respuesta múltiple.

- **nº correctas**: indica cuántas de las alternativas planteadas en el punto anterior deben ser correctas. En función del tipo de preguntas puede ser de sólo una si la pregunta es de respuesta simple o más de una si es de repuesta compuesta o de respuesta abierta.

[1] El valor 1 indica sólo variables binarias, el valor 2 sólo variables genéricas y el valor 3 la combinación aleatoria de ambos tipos.

- **nº incógnitas**: número de valores dentro de la distribución de probabilidad planteada en el enunciado que mostraran ocultas su valor numérico correspondiente, en forma de incógnita [2].

En la tabla A.2 se presentan las distintas instancias definidas en *Siette* para el tipo de pregunta del primer bloque temático:

Tabla A.2: Instancias de preguntas *Siette* para primer bloque temático.

Instancia	clase	precisión	nº variables	tipo	nºalternativas	nº correctas	nº incognitas
1_1	IpRelQuestion1	2	3	1	4	1	
1_2	IpRelQuestion1	2	3	1	4	2	
1_3	IpRelQuestion1	1	4	1	4	2	
2_1	IpRelQuestion2	2	3	1	4	1	
2_2	IpRelQuestion2	2	3	1	4	2	
2_3	IpRelQuestion2	1	4	1	4	2	
3_1	IpRelQuestion3	2	3	1	4	1	
3_2	IpRelQuestion3	2	3	1	4	2	
3_3	IpRelQuestion3	1	2	3	4	2	
4_1	IpRelQuestion4	2	3	1			3
4_2	IpRelQuestion4	2	3	2			2
4_3	IpRelQuestion4	2	3	3			2

A.2. Segundo bloque temático

Las preguntas de los tipos 5 y 6 forman parte del segundo bloque temático y corresponden a preguntas que plantean relaciones de independencia probabilística y de independencia probabilística condicional entre variables aleatorias o nodos de un grafo. En este caso el sistema sobre el que se plantea la pregunta consiste en un grafo dirigido o no dirigido. La parametrización asociada a este tipo de preguntas es la siguiente:

- **nombre**: nombre que recibe la pregunta y el grafo que se plantea en el enunciado. Este nombre será utilizado para nombrar los archivos necesarios que se crean durante la generación de la pegunta.

- **nº nodos**: número de nodos que forman parte del grafo que se plantea en el enunciado.

[2]Este parámetro es aplicable sólo a las preguntas de tipo 4.

- **nº enlaces**: número de aristas o enlaces que constituyen el grafo planteado en el enunciado de la pregunta.

- **condicional**: indica si las relaciones que se generan entre los nodos o variables aleatorias del grafo, incluyen también relaciones de independencia condicional, es decir una relación condicional entre un grupo de variables respecto de otra variable o grupo de variables, por ejemplo la relación $I_p(A, B \mid C)$.

- **resp.abierta:** indica que la respuesta a la pregunta planteada es abierta. Es decir, en lugar de elegir una entre una serie de alternativas propuestas como respuesta a la pregunta planteada, el alumno dará una respuesta corta como solución a la pregunta. Esta consistirá normalmente en un valor numérico, una expresión, etc.

- **nº correctas**: número de respuestas que han de ser correctas entre las alternativas planteadas en el enunciado de la pregunta.

- **nº alternativas**: número de posibles respuestas o alternativas a la pregunta planteada en el enunciado.

Tabla A.3: Instancias de preguntas Siette para segundo bloque temático.

Instancia	clase	nombre	nodos	enlaces	condic.	resp.abierta	correctas	alternativas
5_1	IpRelQuestion5	Pregunta5_1	5	5	false	false	1	4
5_2	IpRelQuestion5	Pregunta5_2	5	5	true	false	1	4
5_22	IpRelQuestion5	Pregunta5_22	*n*	*e*	*a*	false	1	4
5_3	IpRelQuestion5	Pregunta5_3	5	5	false	true	1	4
5_4	IpRelQuestion5	Pregunta5_4	5	5	true	true	1	4
5_44	IpRelQuestion5	Pregunta5_44	*n*	*e*	*a*	true	1	4
5_5	IpRelQuestion5	Pregunta5_5	5	3	false	false	2	4
5_6	IpRelQuestion5	Pregunta5_6	5	3	true	false	2	4
5_66	IpRelQuestion5	Pregunta5_66	*n*	*e*	*a*	false	2	4
6_1	IpRelQuestion6	Pregunta6_1	6	6	false	false	1	4
6_2	IpRelQuestion6	Pregunta6_2	6	6	true	false	1	4
6_22	IpRelQuestion6	Pregunta6_22	*n*	*e*	*a*	false	1	4
6_3	IpRelQuestion6	Pregunta6_3	6	6	false	true	1	4
6_4	IpRelQuestion6	Pregunta6_4	6	6	true	true	1	4
6_44	IpRelQuestion6	Pregunta6_44	*n*	*e*	*a*	true	1	4
6_5	IpRelQuestion6	Pregunta6_5	5	3	false	false	2	4
6_6	IpRelQuestion6	Pregunta6_6	5	3	true	false	2	4
6_66	IpRelQuestion6	Pregunta6_66	*n*	*e*	*a*	false	2	4

donde los valores incógnita de las instancias marcadas en negrita, se obtienen de las siguientes expresiones:

$int\ n = Random.nextInt(2, 10);$

$int\ e = Random.nextInt(2, n);$

$boolean\ a = Random.nextBoolean();$

A.3. Tercer bloque temático

Por último el tercer bloque temático está constituido por las preguntas de tipo 7, 8 y 9. En este bloque se plantean preguntas que tienen que ver con inferencia en redes bayesianas. A partir de la red bayesiana planteada en el enunciado se pregunta por la probabilidad a posteriori de una variable de interés tras introducir cierta evidencia en la red. En la tabla A.4 se especifican las instancias de preguntas de este tipo que se plantean en *Siette*. Las instancias de preguntas de tipo 7 tienen que ver con la red bayesiana *Disease-Test*, las instancias de preguntas de tipo 8 plantean redes bayesianas extraídas de la factoría de *OpenMarkov* [3], mientras que las instancias de preguntas de tipo 9 trabajan con redes bayesianas de generación aleatoria.

Tabla A.4: Instancias de preguntas *Siette* para tercer bloque temático.

Instancia	**clase**	**nombre**	**precisión**	**nº nodos**	**nº enlaces**	**tipo red**	**nº alternativas**
7_1	IBNQuestion7	Pregunta7_1	2				
7_2	IBNQuestion7	Pregunta7_2	4				
8_1	IBNQuestion8	Pregunta8_1	4			0	4
8_2	IBNQuestion8	Pregunta8_2	4			1	4
8_3	IBNQuestion8	Pregunta8_3	4			2	4
8_4	IBNQuestion8	Pregunta8_4	p			r	4
9_1	IBNQuestion9	Pregunta9_1	3	4	3		4
9_2	IBNQuestion9	Pregunta9_2	2	6	5		4
9_3	IBNQuestion9	Pregunta9_3	p	n	e		4

donde los valores incógnita de las instancias marcadas en negrita, se obtienen de las siguientes expresiones:

$int\ p = Random.nextInt(2, 4);$

$int\ r = Random.nextInt(0, 2);$

$int\ n = Random.nextInt(2, 10);$

[3]la factoría *OpenMarkov* ofrece los siguientes tipos de redes: red bN_XYZ, red bN_ABC o red Asia

$int\ e = Random.nextInt(2, n)$;

A partir de la información recogida en las tablas anteriores dispondríamos de todo lo necesario para definir la asignatura *Siette* Modelos Gráficos Probabilistas y los test de autoevaluación de la asignatura que se alimentarían de las instancias de preguntas definidas.

Apéndice B

Manual de usuario

En este capítulo vamos a describir el manual de usuario de la librería de generación automática de preguntas sobre Modelos Gráficos Probabilistas en Inteligencia Artificial. La librería *LgapMgp* desarrollada en este proyecto funciona actualmente bajo la versión 1.7 de Java.

La librería genera preguntas en formato test sobre el tema Modelos Gráficos Probabilistas. Como se ha comentado a lo largo de toda la memoria de este proyecto, se ha dividido la librería en tres bloques temáticos dentro del tema principal. A su vez cada uno de estos tres bloques temáticos da lugar a distintos tipos de pregunta que nos permitirán en última instancia generar una pregunta concreta sobre el bloque temático elegido. La generación de los distintos tipos de pregunta se realiza mediante la instanciación de una determinada clase que genera un tipo de pregunta en particular. Los distintos tipos de pregunta que forman parte del repertorio de la librería de generación automática se presentan en el apéndice A. Cada una de estas clases forma parte de un paquete que se corresponde con un determinado bloque temático.

El primer bloque temático corresponde al paquete llamado *relipDP*. En este bloque se definen cuatro tipos de preguntas que plantean cuestiones sobre relaciones de independencia probabilística e independencia probabilística condicional entre variables aleatorias de un sistema. Los métodos ofrecidos por la clase se pueden dividir en dos grupos:

- un primer grupo formado por los métodos de la interfaz con el sistema *Siette*. Estos métodos permitirán definir en *Siette* cada instancia de pregunta que forma parte de una asignatura. Dependiendo de si se trata de una pregunta de respuesta única o de respuesta múltiple implementaremos la interfaz correspondiente a cada tipo de pregunta. Por ejemplo los métodos implementados para una pregunta de tipo 1 en la clase *IpRelQuestion1* son los correspondientes a la interfaz *RespuestaMultiple* que

Tabla B.1: Métodos de la interfaz con *Siette* de una pregunta de tipo 1 *IpRelQuestion1*

Método	**Descripción**
public String getEnunciado ();	devuelve el enunciado de la pregunta
public Vector <String> getRespuestasCorrectas();	devuelve las n respuestas correctas a la pregunta
public String getRefuerzoRespuestasCorrectas();	devuelve refuerzo asociado respuestas correctas
public Vector <String> getRespuestasIncorrectas();	devuelve n respuestas incorrectas
public String getRefuerzoRespuestasIncorrectas();	devuelve refuerzo asociado a espuesta incorrecta

a su vez hereda del interfaz *PreguntaSiette*. La implementación de ambos interfaces implica la implementación de los siguientes métodos que se reflejan en la tabla B.1

- un segundo grupo formado por el método *genTextFilePreguntas()*, será el encargado de generar la salida de la pregunta generada en un fichero de formato PDF. Para la generación de la salida PDF es necesario que exista en el sistema donde se ejecuta la librería una distribución de *Latex* (LaTeX, 2010) como la distribución *MikTex* que se ha utilizado en este proyecto para la plataforma Windows (Schenk, 2013). Desde la página de *MikTex* se puede realizar la descarga del DVD de instalación. La librería *LgapMgp* genera un fichero en formato *Latex* de extensión .tex, haciendo uso del método *TexToPdf ()* de la clase *Util* del paquete *util* se transforma el fichero *Latex* en un fichero PDF que presenta la pregunta generada. La transformación del archivo .tex com comandos *Latex* a PDF se realiza mediante la invocación al comando **pdflatex** de la distribución de *MikTex*. Un ejemplo de generación de pregunta en este formato lo podemos ver en el apéndice D.

De la misma forma que lo explicado para las preguntas del bloque temático 1, los bloques temáticos 2 y 3 también están agrupados en los paquetes *relipGrafo* y *rb* respectivamente. De esta manera la implementación de los métodos de la interfaz con *Siette* así como la generación de la salida PDF determina un comportamiento común de cualquier tipo de pregunta dentro de un bloque temático que se quiera implementar en un futuro con la ampliación de la librería desarrollada. La implementación de estos métodos garantizan dos de los objetivos principales de este proyecto como son:

- por un lado la integración en el sistema *Siette* como abastecedor de preguntas de test dentro de una asignatura definida en dicho sistema.
- por otro lado la generación de exámenes de evaluación en formato PDF desde un programa cliente de la librería en la plataforma Java.

En la figura B.1 se muestra un ejemplo de uso del tipo de pregunta 1 desde el sistema *Siette* (pestaña "Contenido", ver documentación sobre Siette (Siette, 2013)).

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="es.uned.dia.pfc.lgap.mgp.relipDP.IpRelQuestion1" %>


<%
   IpRelQuestion1 miPregunta1 = new IpRelQuestion1(2, 3, 1, 4, 1);

   String enunciado = miPregunta1.getEnunciado();
   String respuestaCorrecta = miPregunta1.getRespuestasCorrectas().get(0);
   String respuestaIncorrecta1 = miPregunta1.getRespuestasIncorrectas().get(0);
   String respuestaIncorrecta2 = miPregunta1.getRespuestasIncorrectas().get(1);
   String respuestaIncorrecta3 = miPregunta1.getRespuestasIncorrectas().get(2);
   String refuerzoCorrecta = miPregunta1.getRefuerzoRespuestasCorrectas();
   String refuerzoIncorrecta = miPregunta1.getRefuerzoRespuestasIncorrectas();
%>

<%= enunciado %>
```

Figura B.1: Ejemplo de uso pregunta de test *LgapMgp* en *Siette*.

Apéndice C

Manual de instalación

En este apéndice vamos a explicar el modo de proceder para realizar la instalación de la librería atendiendo a dos usos posibles de la misma:

- como parte integrante del sistema *Siette* que abastece de preguntas a la asignatura Modelos Gráficos Probabilistas.
- como API dentro de un proyecto Java que hace uso de la librería para la generación automática de preguntas que formen parte de exámenes de test o ejercicios de evaluación sobre temas relativos a los Modelos Gráficos Probabilistas.

En los siguientes apartados explicamos cada uno de estos usos en detalle.

C.1. Instalación de la librería *LgapMgp* en *Siette*

El entregable principal del presente proyecto es la librería desarrollada, la librería de generación automática de preguntas sobre Modelos Gráficos Probabilistas, *LgapMgp*. El entregable consiste pues en una librería Java empaquetada en un archivo de tipo jar. Para su integración en el sistema *Siette*, este archivo ha de ser empaquetado a su vez en un archivo comprimido de tipo .zip. El primer paso para su uso dentro de *Siette* es el siguiente:

1. dentro de la asignatura Modelos Gráficos Probabilistas, ir a la pestaña de Gestión de archivos.
2. En el campo directorio seleccionar la ruta **5988/WEB-INF/lib.**
3. En la sección "Enviar fichero" pulsar el botón "Examinar" y seleccionar el fichero *LgapMgp71.zip* de la ubicación que corresponda como entregable de este proyecto.

4. Una vez seleccionado el archivo ***LgapMgp71.zip*** de la librería, pulsar el botón "Enviar".

5. Si la operación es correcta, ya tenemos la librería integrada en el sistema *Siette*. El sistema descomprimirá el archivo y se desplegará el "proyecto jar" en el sistema *Siette* para poder hacer uso de las distintas clases correspondientes a los tipos de pregunta que forman parte del repertorio ofrecido por la librería.

A partir de este momento tendremos disponibles bajo el esquema de generación JSP la definición de los distintos tipos de preguntas o instancias que se pueden generar en *Siette* dentro de la asignatura Modelos Gráficos Probabilistas. Ver apéndice A para las distintas opciones de parametrización de cada clase que dan lugar a las distintas variaciones sobre las distintas instancias de preguntas que se pueden plantear y el apéndice B para el uso de las preguntas ofrecidas por la librería.

La figura muestra el despliegue de la librería desarrollada en el sistema *Siette*.

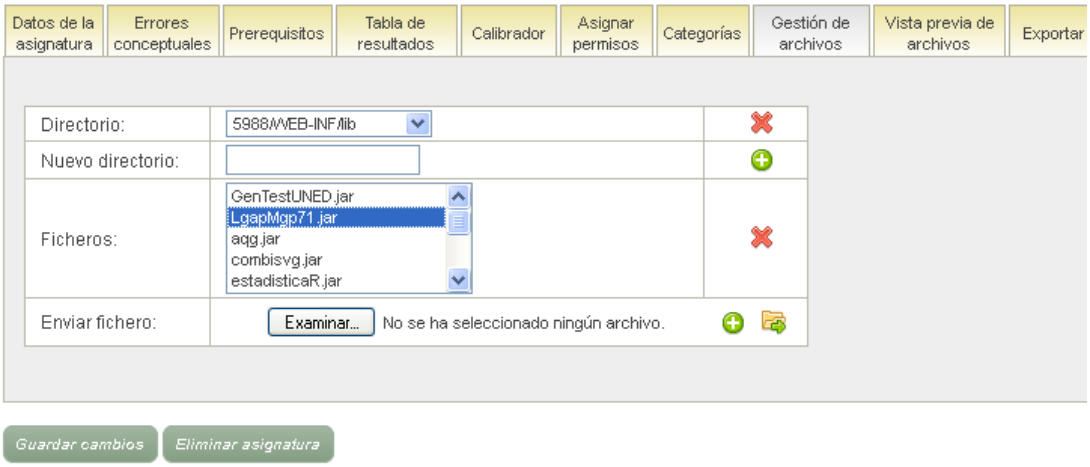


Figura C.1: Despliegue librería *LgapMgp* en sistema *Siette*.

C.2. Instalación de la librería *LgapMgp* como API

El uso de la librería desarrollada como API se realizará normalmente desde un entorno integrado de desarrollo (IDE) como puede ser por ejemplo Eclipse. Dicho entorno de desarrollo es el que ha sido utilizado en este proyecto para el desarrollo de la librería de generación automática de preguntas. Un programa cliente o proyecto Java que haga uso de la librería desarrollada en este proyecto deberá incluir la misma como librería

externa (*LgapMgp71.jar*) al proyecto y hacer uso de las distintas clases instanciando adecuadamente cada una de ellas, ver manual de usuario B para el uso correcto de las clases ofrecidas por la librería.

Los CD-ROM proporcionados junto con la memoria de este proyecto contienen una máquina virtual donde se incorpora el entorno de desarrollo IDE utilizado para la implementación de esta librería.

Apéndice D

Ejemplos de test generados

En este apéndice vamos a mostrar como ejemplos algunos de los tipos de preguntas de generación automática que se pueden generar con la librería de generación automática de preguntas desarrollada en este proyecto de fin de carrera sobre Modelos Gráficos Probabilistas en Inteligencia Artificial.

En los siguientes apartados, mostraremos un ejemplo representativo de cada uno de los bloques temáticos en los que se divide el temario de la librería. Presentaremos tanto el enunciado completo de la pregunta como las distintas alternativas o respuestas posibles que se pueden dar a la misma al tratarse de una pregunta de test, en la que el alumno debe elegir como respuesta correcta una de las propuestas. Además en cada ejemplo se da la información de refuerzo que permite al alumno conocer cuáles son los pasos que tiene que dar para llegar a la solución de la pregunta planteada (refuerzo positivo) y en el caso de las respuestas incorrectas el motivo por el que son incorrectas (refuerzo negativo). Esta información de refuerzo se presenta junto con la respuesta correcta marcada como solución a la pregunta.

El hecho de marcar o no la respuesta correcta, mostrar o no la información de refuerzo es algo configurable en el momento de generar cada pregunta, concretamente en los parámetros segundo y tercero del método *genPDFLatexFile()* de la clase *ExamenTest.* Cada uno de los ejemplos presentados en los siguientes apartados corresponden a la ejecución del correspondiente test en JUnit que instancia una clase para generar un tipo de pregunta, incorpora la pregunta generada a la instancia de la clase *ExamenTest* y realiza la invocación al método *genPDFLatexFile()* para la generación de un fichero en formato PDF con el resultado de la pregunta planteada, marcando la respuesta correcta y proporcionando la información de refuerzo tanto para las respuestas correctas como para las respuestas incorrectas.

D.1. Ejemplo de pregunta del bloque temático 1

En este apartado presentamos un ejemplo de pregunta generada sobre el bloque temático 1 resultado de la ejecución del test de JUnit *TestIpRelQuestion1*. Se plantea una pregunta en la que se presentan tres variables aleatorias binarias y se trabaja con una precisión de dos decimales en los cálculos. Las alternativas o posibles respuestas a la pregunta planteada serán cuatro y tan sólo una de ellas es la respuesta correcta. El código Java correspondiente al test JUnit que da lugar a la pregunta se puede ver en la figura D.1

```
int precision, numVar, TipoVar, NRespuestas, NCorrectas;
precision = 2;
numVar = 3;
TipoVar = 1;      // (1 -> binarias, 2-> aleatorias normales, 3-> de ambas)
NRespuestas = 4; // Respuestas propuestas totales
NCorrectas = 1;  // Número mínimo de respuestas correctas entre las propuestas.

// Salida a PDF.
try {

    myQuestion1 = new IpRelQuestion1(precision, numVar, TipoVar, NRespuestas, NCorrectas);
    examen = ExamenTest.getInstance();
    examen.addQuestions(myQuestion1);
    examen.genPDFLatexFile("Pregunta1", true, true);

} catch (LgapException e) {
    e.printStackTrace();
}
```

Figura D.1: Código Java para la generación de una pregunta de tipo 1.

A continuación se presenta el resultado de la ejecución que da lugar a la pregunta:

Ejercicio 1.- Sea una distribución de probabilidad P dada por la siguiente tabla:

$A \quad B \quad C$	$P(A,B,C)$
$(+a,+b,+c)$	0.02784
$(+a,+b,\neg c)$	0.01768
$(+a,\neg b,+c)$	0.11136
$(+a,\neg b,\neg c)$	0.08632
$(\neg a,+b,+c)$	0.06816
$(\neg a,+b,\neg c)$	0.07072
$(\neg a,\neg b,+c)$	0.27264
$(\neg a,\neg b,\neg c)$	0.34528

Señale cuál de las siguientes relaciones es verdadera:

1.- $I_p(B,C)$ FALSO

2.- $I_p(A,B)$ FALSO

3.- $I_p(B,C|A)$ FALSO

4.- $I_p(A,B|C)$ VERDADERO

A continuación se muestra el refuerzo para cada tipo de pregunta:

Solución a pregunta 1 (pregunta tipo 1)

1.- $I_p(B,C)$ FALSO

2.- $I_p(A,B)$ FALSO

3.- $I_p(B,C|A)$ FALSO

4.- $I_p(A,B|C)$ VERDADERO

Refuerzo correctas:

Comprobamos la independencia condicional entre las variables de la expresión: $I_p\left(A, B|C\right)$.
Para cada configuración obtenida comprobrar si se cumple la condición de independencia probabilistica siguiente:

$$P\left(A, B|C\right) = P\left(A|C\right) \times P\left(B|C\right)$$

1ª Combinación

$$P\left(+a, +b| + c\right) = P\left(+a| + c\right) \times P\left(+b| + c\right)$$

$$\frac{P\left(+a, +b, +c\right)}{P\left(+c\right)} = \frac{P\left(+a, +c\right)}{P\left(+c\right)} \times \frac{P\left(+b, +c\right)}{P\left(+c\right)}$$

$$\frac{0.02784}{0.48} = \frac{0.1392}{0.48} \times \frac{0.096}{0.48}$$

$$0.06 = 0.29 \times 0.2$$

$$0.06 = 0.06$$

Se cumple la propiedad para esta combinación

2ª Combinación

$$P\left(+a, +b|\neg c\right) = P\left(+a|\neg c\right) \times P\left(+b|\neg c\right)$$

$$\frac{P\left(+a, +b, \neg c\right)}{P\left(\neg c\right)} = \frac{P\left(+a, \neg c\right)}{P\left(\neg c\right)} \times \frac{P\left(+b, \neg c\right)}{P\left(\neg c\right)}$$

$$\frac{0.01768}{0.52} = \frac{0.104}{0.52} \times \frac{0.0884}{0.52}$$

$$0.03 = 0.2 \times 0.17$$

$$0.03 = 0.03$$

Se cumple la propiedad para esta combinación

3ª Combinación

$$P\left(+a, \neg b| + c\right) = P\left(+a| + c\right) \times P\left(\neg b| + c\right)$$

$$\frac{P\left(+a, \neg b, +c\right)}{P\left(+c\right)} = \frac{P\left(+a, +c\right)}{P\left(+c\right)} \times \frac{P\left(\neg b, +c\right)}{P\left(+c\right)}$$

$$\frac{0.11136}{0.48} = \frac{0.1392}{0.48} \times \frac{0.384}{0.48}$$

$$0.23 = 0.29 \times 0.8$$

$$0.23 = 0.23$$

Se cumple la propiedad para esta combinación

4ª Combinación

$$P\left(+a,\neg b|\neg c\right)=P\left(+a|\neg c\right)\times P\left(\neg b|\neg c\right)$$

$$\frac{P\left(+a,\neg b,\neg c\right)}{P\left(\neg c\right)}=\frac{P\left(+a,\neg c\right)}{P\left(\neg c\right)}\times\frac{P\left(\neg b,\neg c\right)}{P\left(\neg c\right)}$$

$$\frac{0.08632}{0.52}=\frac{0.104}{0.52}\times\frac{0.4316}{0.52}$$

$$0.17=0.2\times0.83$$

$$0.17=0.17$$

Se cumple la propiedad para esta combinación

5ª Combinación

$$P\left(\neg a,+b|+c\right)=P\left(\neg a|+c\right)\times P\left(+b|+c\right)$$

$$\frac{P\left(\neg a,+b,+c\right)}{P\left(+c\right)}=\frac{P\left(\neg a,+c\right)}{P\left(+c\right)}\times\frac{P\left(+b,+c\right)}{P\left(+c\right)}$$

$$\frac{0.06816}{0.48}=\frac{0.3408}{0.48}\times\frac{0.096}{0.48}$$

$$0.14=0.71\times0.2$$

$$0.14=0.14$$

4

Se cumple la propiedad para esta combinación

6ª Combinación

$$P\left(\neg a, +b|\neg c\right) = P\left(\neg a|\neg c\right) \times P\left(+b|\neg c\right)$$

$$\frac{P\left(\neg a, +b, \neg c\right)}{P\left(\neg c\right)} = \frac{P\left(\neg a, \neg c\right)}{P\left(\neg c\right)} \times \frac{P\left(+b, \neg c\right)}{P\left(\neg c\right)}$$

$$\frac{0.07072}{0.52} = \frac{0.416}{0.52} \times \frac{0.0884}{0.52}$$

$$0.14 = 0.8 \times 0.17$$

$$0.14 = 0.14$$

Se cumple la propiedad para esta combinación

7ª Combinación

$$P\left(\neg a, \neg b| + c\right) = P\left(\neg a| + c\right) \times P\left(\neg b| + c\right)$$

$$\frac{P\left(\neg a, \neg b, +c\right)}{P\left(+c\right)} = \frac{P\left(\neg a, +c\right)}{P\left(+c\right)} \times \frac{P\left(\neg b, +c\right)}{P\left(+c\right)}$$

$$\frac{0.27264}{0.48} = \frac{0.3408}{0.48} \times \frac{0.384}{0.48}$$

$$0.57 = 0.71 \times 0.8$$

$$0.57 = 0.57$$

Se cumple la propiedad para esta combinación

8ª Combinación

$$P(\neg a, \neg b|\neg c) = P(\neg a|\neg c) \times P(\neg b|\neg c)$$

$$\frac{P(\neg a, \neg b, \neg c)}{P(\neg c)} = \frac{P(\neg a, \neg c)}{P(\neg c)} \times \frac{P(\neg b, \neg c)}{P(\neg c)}$$

$$\frac{0.34528}{0.52} = \frac{0.416}{0.52} \times \frac{0.4316}{0.52}$$

$$0.66 = 0.8 \times 0.83$$

$$0.66 = 0.66$$

Se cumple la propiedad para esta combinación

Se cumple para todas las combinaciones.
Concluimos que se cumple la propiedad $I_p(A, B|C)$.

Refuerzo incorrectas:

Comprobando la independencia entre las variables $I_p(B, C)$.
Probabilidades marginales:

$$P(+b) = 0.1844$$

$$P(\neg b) = 0.8156$$
$$P(+c) = 0.48$$
$$P(\neg c) = 0.52$$

Configuraciones obtenidas de las variables a comprobrar con sus respectivas probabilidades:

$$P(+b, +c) = 0.096$$
$$P(+b, \neg c) = 0.0884$$
$$P(\neg b, +c) = 0.384$$
$$P(\neg b, \neg c) = 0.4316$$

Para cada configuración obtenida comprobrar si se cumple la condición de independencia probabilistica:

- No se verifica la propiedad para la combinación $(+b, +c)$ pues NO se cumple:
$$P(+b, +c) = P(+b) \times P(+c)$$
$$0.096 \neq 0.0885$$
- No se verifica la propiedad para la combinación $(+b, \neg c)$ pues NO se cumple:
$$P(+b, \neg c) = P(+b) \times P(\neg c)$$
$$0.0884 \neq 0.0959$$
- No se verifica la propiedad para la combinación $(\neg b, +c)$ pues NO se cumple:
$$P(\neg b, +c) = P(\neg b) \times P(+c)$$
$$0.384 \neq 0.3915$$
- No se verifica la propiedad para la combinación $(\neg b, \neg c)$ pues NO se cumple:
$$P(\neg b, \neg c) = P(\neg b) \times P(\neg c)$$
$$0.4316 \neq 0.4241$$

Comprobando la independencia entre las variables $I_p(A, B)$.
Probabilidades marginales:

$$P(+a) = 0.2432$$

$$P(\neg a) = 0.7568$$

$$P(+b) = 0.1844$$

$$P(\neg b) = 0.8156$$

Configuraciones obtenidas de las variables a comprobrar con sus respectivas probabilidades:

$$P(+a, +b) = 0.04552$$

$$P(+a, \neg b) = 0.19768$$

$$P(\neg a, +b) = 0.13888$$

$$P(\neg a, \neg b) = 0.61792$$

Para cada configuración obtenida comprobrar si se cumple la condición de independencia probabilistica:

- No se verifica la propiedad para la combinación $(+a, +b)$ pues NO se cumple:

$$P(+a, +b) = P(+a) \times P(+b)$$

$$0.04552 \neq 0.0448$$

- No se verifica la propiedad para la combinación $(+a, \neg b)$ pues NO se cumple:

$$P(+a, \neg b) = P(+a) \times P(\neg b)$$

$$0.19768 \neq 0.1984$$

- No se verifica la propiedad para la combinación $(\neg a, +b)$ pues NO se cumple:

$$P(\neg a, +b) = P(\neg a) \times P(+b)$$

$$0.13888 \neq 0.1396$$

- No se verifica la propiedad para la combinación $(\neg a, \neg b)$ pues NO se cumple:

$$P(\neg a, \neg b) = P(\neg a) \times P(\neg b)$$

$$0.61792 \neq 0.6172$$

Comprobamos la independencia condicional entre las variables de la expresión: $I_p(B, C|A)$.
Para cada configuración obtenida comprobrar si se cumple la condición de independencia probabilistica siguiente:

$$P(B, C|A) = P(B|A) \times P(C|A)$$

1ª Combinación

$$P(+b, +c| + a) = P(+b| + a) \times P(+c| + a)$$

$$\frac{P(+b, +c, +a)}{P(+a)} = \frac{P(+b, +a)}{P(+a)} \times \frac{P(+c, +a)}{P(+a)}$$

$$\frac{0.02784}{0.2432} = \frac{0.04552}{0.2432} \times \frac{0.1392}{0.2432}$$

$$0.11 = 0.19 \times 0.57$$

$$0.11 = 0.11$$

Se cumple la propiedad para esta combinación

2ª Combinación

$$P\left(+b,+c|\neg a\right)=P\left(+b|\neg a\right)\times P\left(+c|\neg a\right)$$

$$\frac{P\left(+b,+c,\neg a\right)}{P\left(\neg a\right)}=\frac{P\left(+b,\neg a\right)}{P\left(\neg a\right)}\times\frac{P\left(+c,\neg a\right)}{P\left(\neg a\right)}$$

$$\frac{0.06816}{0.7568}=\frac{0.13888}{0.7568}\times\frac{0.3408}{0.7568}$$

$$0.09=0.18\times0.45$$

$$0.09\neq0.08$$

NO se cumple la propiedad para esta combinación

Por tanto no se cumple la propiedad de independencia probabilística condicional de las variables $I_p\left(B,C|A\right)$.

D.2. Ejemplo de pregunta del bloque temático 2

En este apartado vamos a presentar un ejemplo de generación de pregunta del segundo bloque temático correspondiente a las relaciones de independencia probabilística e independencia probabilística condicional sobre grafos. El resultado de la generación corresponde al código Java que se ejecuta correspondiente al test JUnit *TestIpRelQuestion6* y que se puede ver en la figura D.2. En este caso generamos una pregunta que presenta un grafo de 5 nodos y 6 enlaces. La pregunta incluye relaciones de independencia probabilística entre dos variables o nodos condicionada a una tercera variable (esto se indica con el valor 'true' para el cuarto parámetro del constructor de la clase *IpRelQuestion6*). Es una pregunta que presenta cuatro alternativas o posibles respuestas a la pregunta planteada y sólo una de ellas es la respuesta correcta.

```
try {
    int nNodos = 5;
    int nEnlaces = 6;
    int NRespuestas = 4; // Respuestas propuestas totales
    int NCorrectas = 1;  // Número mínimo de respuestas correctas entre las propuestas.

    ipRelQuestion6 = new IpRelQuestion6("pregunta6", nNodos, nEnlaces,
                                        true, false,
                                        NCorrectas, NRespuestas);

    // Salida a PDF.
    examen = ExamenTest.getInstance();
    examen.addQuestions(ipRelQuestion6);
    examen.genPDFLatexFile(ipRelQuestion6.getNombre(), true, true);

} catch (LgapException e1) {
    e1.printStackTrace();
    fail(e1.getMessage());
}
```

Figura D.2: Código Java para la generación de una pregunta de tipo 6.

Seguidamente se presenta el resultado de la ejecución que da lugar a la pregunta.

Ejercicio 1.- Sea un grafo no dirigido G que contiene 5 nodos A, B, C, D, E y los siguientes enlaces: $A \rightarrow C, A \rightarrow E, B \rightarrow C, B \rightarrow E, C \rightarrow A, E \rightarrow C$. Indique cuáles de las siguientes relaciones son verdaderas:

1.- $\neg I_G(C, D|A)$ FALSO

2.- $I_G(B, C|E)$ FALSO

3.- $I_G(C, E|D)$ FALSO

4.- $\neg I_G(A, C|E)$ VERDADERO

A continuación se muestra el refuerzo para cada tipo de pregunta:

Solución a pregunta 1 (pregunta tipo 6)

Solución

1.- $\neg I_G(C, D|A)$ FALSO

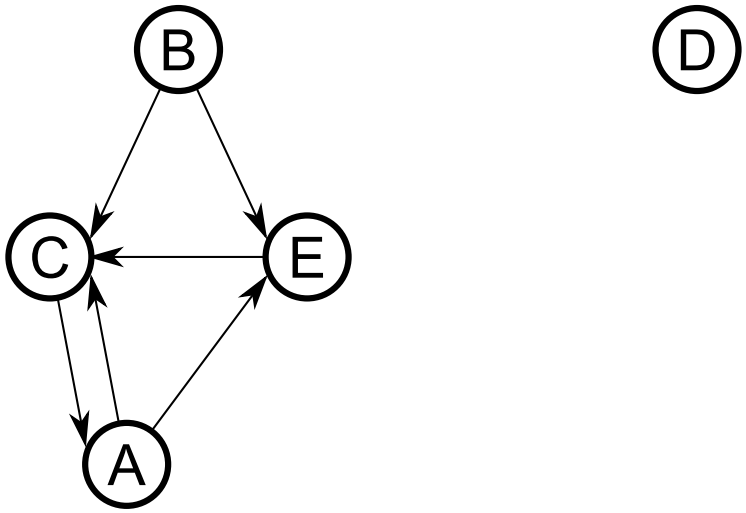
2.- $I_G(B, C|E)$ FALSO

3.- $I_G(C, E|D)$ FALSO

4.- $\neg I_G(A, C|E)$ VERDADERO

Refuerzo:

Observando el grafo G :



A continuación verificamos si se cumplen cada una de las relaciones. Calculamos cada uno de los caminos activos que existen entre las variables indicadas en la relación

1.- $\neg I_G\,(C, D|A)$ FALSO
CAMINOS ACTIVOS = []
CAMINOS BLOQUEADOS = []

2.- $I_G\,(B, C|E)$ FALSO
CAMINOS ACTIVOS = [[B, C]]
CAMINOS BLOQUEADOS = [[B, E, C]]

3.- $I_G\,(C, E|D\,)$ FALSO
CAMINOS ACTIVOS = [[C, A, E], [C, B, E], [C, E]]
CAMINOS BLOQUEADOS = []

4.- $\neg I_G\,(A, C|E\,)$ VERDADERO
CAMINOS ACTIVOS = [[A, C]]
CAMINOS BLOQUEADOS = [[A, E, C]]

D.3. Ejemplo de pregunta del bloque temático 3

Por último en este tercer apartado el ejemplo presentado corresponde a una pregunta del tercer bloque temático. En este caso, la pregunta elegida es una de tipo 9 en la que se genera una red bayesiana de manera aleatoria. Para ello será necesario instanciar la clase *iBNQuestion9*. Se describe la red bayesiana generada proporcionando la información correspondiente a la parte cuantitativa de la red, los potenciales de cada nodo y se completa la descripción de la misma con su estructura gráfica, el grafo que la representa, es decir su parte cualitativa. A partir de ahí se introducen una serie de hallazgos en algún nodo de la red y la pregunta consiste en determinar el valor predictivo de algún nodo de la red tras la evidencia introducida.

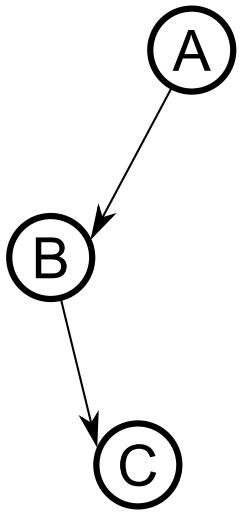
En este caso generamos una pregunta que presenta una red bayesiana formada por un número de nodos comprendido entre dos y cinco, el número de enlaces será del número de nodos menos uno. Se trabaja con una precisión de tres decimales en los cálculos y el número de alternativas o posibles respuestas a la pregunta planteada será de cuatro siendo sólo una de ellas es la respuesta correcta. El código Java correspondiente al test JUnit *TestIBNQuestion9* que da lugar a la pregunta se puede ver en la figura D.3.

```
IBNQuestion9 iBNQuestion9;
ExamenTest examen;

try {

    int nNodes = 2 + (int) Util.getRandomNum(5);
    int precision = 3;
    int numRespuestas = 4;
    iBNQuestion9 = new IBNQuestion9("pregunta9", nNodes, nNodes-1,
                                    precision, numRespuestas);

    // Salida a PDF.
    examen = ExamenTest.getInstance();
    examen.addQuestions(iBNQuestion9);
    examen.genPDFLatexFile(iBNQuestion9.getNombre(), true, true);

} catch (LgapException e1) {
    e1.printStackTrace();
    fail(e1.getMessage());
}
```

Figura D.3: Código Java para la generación de una pregunta de tipo 9.

Seguidamente se presenta el resultado de la ejecución que da lugar a la pregunta.

Ejercicio 1.- Sea la red bayesiana de la figura:



Con los siguientes estados para cada una de las variables o nodos de la red:
$A = (yes, no)$
$B = (positive, negative)$
$C = (positive, negative)$

Las tablas de probabilidades condicionadas que definen la red son las siguientes:
$P(A) = \{0.383, 0.617\}$
$P(B|A) = \{0.2, 0.8, 0.126, 0.874\}$
$P(C|B) = \{0.099, 0.901, 0.191, 0.809\}$

Si se introduce en la variable B el hallazgo ″negative″.
¿Cuál es la probabilidad asociada al valor ″yes″ de la variable A?

1.- 0.203 FALSO

2.- 0.545 FALSO

3.- 0.196 FALSO

4.- 0.362 VERDADERO

A continuación se muestra el refuerzo para cada tipo de pregunta:

Solución a pregunta 1 (pregunta tipo 9)

A partir de las tablas de probabilidades condicionadas dadas en el enunciado, obtenemos la siguiente distribución de probabilidad:

$A \quad B \quad C$	$P(A,B,C)$
$(+a,+b,+c)$	0.008
$(+a,+b,\neg c)$	0.059
$(+a,\neg b,+c)$	0.069
$(+a,\neg b,\neg c)$	0.248
$(\neg a,+b,+c)$	0.008
$(\neg a,+b,\neg c)$	0.103
$(\neg a,\neg b,+c)$	0.07
$(\neg a,\neg b,\neg c)$	0.436

Las probabilidades a posteriori de cada variable, calculadas a partir de la tabla dada son:

$A \rightarrow \mathrm{P}(A) = \{0.383, 0.617\}$
$B \rightarrow \mathrm{P}(B) = \{0.154, 0.846\}$
$C \rightarrow \mathrm{P}(C) = \{0.177, 0.823\}$

que se obtienen de la proyección de cada valor de cada variable.
La probabilidad asociada al valor "yes" de la variable A vendrá dada por el cociente:

$$P(+a) = \frac{proyección\,(\neg b, +a)}{proyección\,(\neg b)}$$

Ahora si tomamos la proyección de $(\neg b, +a)$ sobre la distribución obtenemos 0.317

Si tomamos la proyección de $(\neg b)$ en la distribución de probabilidad dada, obtenemos el valor 0.823
El resultado final se obtiene como el cociente de ambas

$$0.317/0.823 = 0.385$$

Apéndice E

Listado de siglas, abreviaturas y acrónimos

ANA	Análisis.
API	Application Programming Interface, Interfaz de Programación de Aplicación.
CTT	Teoría Clásica de Test.
DIS	Diseño.
DOC	Documentación.
EEES	Espacio Europeo de Educación Superior.
FUN	Funcional.
GDA	Grafo Dirigido Acíclico.
GNA	Grafo No dirigido Acíclico.
HTML	HyperText Markup Language, Lenguaje de Marcas de HiperTexto.
IDE	Integrated Development Environment, Entorno de Desarrollo Integrado.
IMP	Implementación.
IMPL	Implantación.
JSP	JavaServer Pages, Páginas Servidor Java.
LGAP	Librería de Generación Automática de Preguntas.

LMS	Learning Management System, Sistema Gestor de Contenidos Educativos.
MGP	Modelos Gráficos Probabilistas.
OMG	Object Management Group, Grupo de Gestión de Objetos.
OO	Orientación a Objetos.
PDF	Portable Document Format, Formato Portable de Documento.
PFC	Proyecto de Fin de Carrera.
PRU	Pruebas.
POO	Programación Orientada a Objetos.
REQ	Requisito.
SIETTE	SIstema de Evaluación inTeligenTe mediante tEst.
SMIL	Synchronized Multimedia Integration Language, Lenguaje de Integración Multimedia Sincronizada.
STI	Sistema Tutor Inteligente.
SVG	Scalable Vector Graphics, Gráficos Vectoriales Redimensionables.
SW	Software.
TAI	Tests Adaptativos Informatizados.
TRI	Teoría de Respuesta al Ítem.
UML	Unified Modeling Language, Lenguaje Unificado de Modelado.
UNED	Universidad Nacional de Educación a Distancia.
VPP	Valor Predictivo Positivo.
VPN	Valor Predictivo Negativo.
XML	eXtensible Markup Language, Lenguaje de Marcas eXtensible.

Bibliografía

Agustín, G. C. (2003). *Gestión del Proceso Software*. Editorial Centro de Estudios Ramón Areces, S.A.

Apache (2011). Tomcat. `http://tomcat.apache.org/`. (accessed, 2011).

ArgoUML. Linus tolke. `http://argouml.tigris.org/`.

Bass, L., Clements, P., and Kazman, R. (1997). *Software Architecture in Practice*. Addison-Wesley Professional, Boston, MA.

bitbucket. Bitbucket. `https://bitbucket.org`.

C.Martin, R. (2004). *UML para Programadores Java*. Pearson Prentice Hall.

Daniel Borrajo Millán, Jesús González Boticario, P. I. V. (2006). *Aprendizaje Automático*. Sanz y Torres.

de Miguel Díaz, M. (2007). *Cambio de paradigma metodológico en la educación superior. Exigencias que conlleva Internet.* Universidad de Oviedo.

F.J.Díez (2007,2010). Introducción a los modelos gráficos probabilistas. Available on Internet.

F.J.Díez (2013). Probabilidad y teoría de la decisión. Available on Internet.

Foundation, T. E. Eclipse. `http://www.eclipse.org/`.

Fowler, M. (2005). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (2005). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, MA.

Ide, J. S., Cozman, F. G., and Ramos, F. T. Generating random bayesian networks with constraints on inducedwidth. *Computer*.

Jesús González Boticario, E. G. V. (2003). *Sistemas Interactivos de Enseñanza/Aprendizaje*. Sanz y Torres, Madrid, ES.

J.Mira, A.E.Delgado, J. F. (2001). *Aspectos Básicos de la Inteligencia Artificial*. Sanz y Torres.

LaTeX (2010). Latex: A document preparation system. Available on Internet.

Lyx. lyx the document processor. `http://www.lyx.org//`.

MathJax (2011). Mathjax: display engine for mathematics. Available on Internet.

McGrayne, S. B. (2012). *La teoría que nunca murió. De cómo la regla de Bayes permitió descifrar el código Enigma, perseguir los submarinos rusos y emerger triunfante de dos siglos de controversia*. Drakontos.

Schenk, C. (2013). Miktex. `http://miktex.org/`.

Shalloway, A. and Trott, J. R. (2004). *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Addison-Wesley, Boston, MA.

Siette (2013). Siette: Sistema de evaluación inteligente mediante tests. Available on Internet.

Sommerville, I. (2005). *Ingeniería del software*. seventh edition.

Source, O. inkscape. `http://inkscape.org/`.

S.Pressman, R. (2003). *Ingeniería del software. Un enfoque práctico*. Mc Graw Hill.

UC3M (2009). "Example 1 of the distribution function." Windows Media Video file. `http://homer.uc3m.es/audiovisuales/0809/Octubre/Ej_1_vad.wmv`. (accessed May 11, 2013).

UNED (2013). Openmarkov: Centro de investigación sobre sistemas inteligentes de ayuda a la decisión universidad nacional de educación a distancia (uned). `https://bitbucket.org/cisiad/org.openmarkov/wiki/Home`.