

Universidad de las Ciencias Informáticas

Facultad 3



Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Título:

Clasificación de registros de eventos para la recomendación de algoritmos de descubrimiento de procesos.

Autor(es): Raciél Roche Escobar.

Michel Hernández Pajaro.

Tutor(es): Dr. Raykenler Yzquierdo Herrera.

Ing. Damián Pérez Alfonso.

Junio de 2014

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Raciel Roche Escobar.

Dr. Raykenler Yzquierdo Herrera

Firma del Autor

Firma del Tutor

Michel Hernandez Pajaro.

Ing. Damián Pérez Alfonso.

Firma del Autor

Firma del Tutor

AGRADECIMIENTOS

A todos los profesores que de uno u otra forma contribuyeron a nuestra formación como ingenieros. A los tutores, por su paciencia y dedicación. A los miembros del tribunal y al oponente que influyeron en el perfeccionamiento de esta investigación.

Raciel: Quiero agradecerle a mis padres, en especial a mi madre por todo lo que ella representa y por todo su apoyo, a mis hermanos, a mis abuelos y toda mi familia que siempre me apoyó para lograr esta meta. Le agradezco también a Radel, por ayudarme todo estos años, las libras que tengo te las debo a ti. A mis compañeros de aula, en especial a Michel. En fin a todos los que de una forma u otra me ayudaron a cumplir este trabajo.

Michel: Quiero agradecer en primer lugar a mi madre y mi padre, a mi hermana, mi abuela, mis tíos, a mi primo, en fin a toda mi familia, a mi novia que ya es parte de mi familia, a mis amigos en especial a Raciel. A mis compañeros de aula. A todos los que me ayudaron a terminar la carrera y a terminar esta Trabajo. En fin agradecimientos a todos los que le interese saber que estoy agradecido y que en algún momento estuvieron cerca.

DEDICATORIA

Queremos dedicarles esta tesis a todas las personas que de una forma u otra nos ayudaron a que fuera posible.

RESUMEN

El descubrimiento es un área de la Minería de Procesos donde se utilizan técnicas para tomar un registro de eventos y producir un modelo de procesos. La necesidad de que este modelo represente la ejecución real de los procesos descubiertos, ha impulsado la creación de varios algoritmos de descubrimiento de procesos.

Un algoritmo de descubrimiento puede obtener excelentes resultados frente un registro de eventos dado y sin embargo mostrar resultados de baja calidad frente a otro registro de eventos. Por lo que se debe seleccionar para cada registro de eventos, el algoritmo de descubrimiento que mejor se ajuste a sus características. Una selección inadecuada del algoritmo de descubrimiento para un registro de eventos, conduce a demoras provocadas por su tiempo de ejecución y ajuste, obtención de modelos de baja calidad e interpretaciones inadecuadas del proceso de minado.

Con el fin de identificar los algoritmos de descubrimiento que permitan obtener modelos de mejor calidad para ciertos registros de eventos, este trabajo propuso la clasificación de registros de eventos para la recomendación de algoritmos de descubrimiento de procesos. Se implementó un complemento para ProM que redujo considerablemente el tiempo necesario para la recomendación de algoritmos de descubrimiento, sin afectar la efectividad de la selección en relación a la brindada por la Evaluación Empírica.

PALABRAS CLAVES: Algoritmos de clasificación, algoritmo de descubrimiento, clasificación, procesos, registro de eventos.

TABLA DE CONTENIDOS

AGRADECIMIENTOS	2
DEDICATORIA	3
RESUMEN	4
TABLA DE CONTENIDOS	5
INTRODUCCIÓN	10
1. FUNDAMENTOS TEÓRICOS.....	14
1.1. Minería de Procesos.	14
1.2. Técnicas para la selección de los algoritmos de descubrimiento.....	15
1.3. Recomendación como un problema de Clasificación.....	17
1.4. Minería de Datos.....	18
1.5. Algoritmos de Clasificación.....	20
1.6. Metodología de desarrollo de software.....	26
1.7. Lenguajes de modelado y desarrollo.....	29
1.8. Herramientas	30
1.9. Conclusiones Parciales.....	33
2. PRUEBAS A LOS CLASIFICADORES, PLANIFICACIÓN Y DISEÑO	34
2.1. Preparación y análisis de los datos.....	34
2.2. Configuración y Métodos de Evaluación.....	36
2.3. Resumen de los resultados de las pruebas.....	37
2.4. Conclusión de las pruebas a los clasificadores.....	40
2.5. Ventajas del uso del complemento para la clasificación.....	41
2.6. Propuesta de sistema a desarrollar.....	43
2.7. Modelo conceptual.....	44
2.8. Requisitos del software.....	45
2.9. Requisitos Funcionales.....	45
2.10. Requisitos no funcionales.....	45
2.11. Descripción de los actores del sistema a automatizar.....	46
2.12. Descripción de las historias de usuarios (HU).....	46
2.13. Plan de Iteraciones	48

2.14. Tarjetas CRC.....	48
2.15. Diagrama de clases y paquetes.....	49
2.16. Arquitectura del sistema.....	55
2.17. Patrones de diseño.....	55
2.18. Conclusiones Parciales.....	59
3. IMPLEMENTACIÓN Y PRUEBA	60
3.1. Implementación.....	60
3.2. Estándar de codificación.....	60
3.2.1. Ejemplos de código implementado.....	61
3.3. Desarrollo de las iteraciones.....	62
3.4. Pruebas a desarrollar.....	64
3.4.1. Pruebas Unitarias.....	64
3.4.2. Pruebas de Aceptación.....	65
3.5. Validación de la propuesta de solución.....	67
3.5.1. Validación respecto a Weka.....	67
3.5.2. Validación respecto a la Evaluación Empírica.....	68
3.6. Conclusiones Parciales.....	73
4. CONCLUSIONES.....	75
5. RECOMENDACIONES	76
6. BIBLIOGRAFÍA.....	77
7. ANEXOS.....	81
7.1. Anexo 1: Historias de Usuario	81
7.2. Anexo 2: Tarjetas CRC	89
7.3. Anexo 3: Tareas de ingeniería	91
7.4. Anexo 4: Casos de Prueba de Aceptación.....	100
7.5. Anexo 5: Registros de eventos tomados para la validación respecto a la Evaluación Empírica.....	109
7.6. Anexo 6: Pruebas Unitarias	110
8. GLOSARIO.....	113

ÍNDICE DE FIGURAS.

Figura 1: Representación de las categorías de Minería de Datos.	19
Figura 2: Neurona.....	25
Figura 6: Características de la Estación de Trabajo.....	37
Figura 7: Modelo conceptual.	44
Figura 8: Diagrama de paquetes del sistema.	50
Figura 9: Diagrama de clases del paquete Database.	51
Figura 10: Diagrama de clases del paquete Recommendation.	52
Figura 11: Diagrama de clases del paquete Training.....	53
Figura 12: Diagrama de paquetes.	54
Figura 14: Resultados de las pruebas a las clases OpenCSVFile y DatabaseConfigurationPlugin.....	65
Figura 19: Resumen de las no conformidades detectadas.	67
Figura 14: Resultados de las pruebas a las clases OpenCSVFile y DatabaseConfigurationPlugin.....	110
Figura 15: Resultados de las pruebas a las clases ClasificationPlugin y ClasificationAlgorithm.	111
Figura 16: Resultados de las pruebas a las clases ClasificationTrainingVisualization y ClasificationTrainingPlugin.....	111
Figura 17: Resultados de las pruebas a la clase RecommendationVisualization.	111
Figura 18: Resultados de las pruebas a las clases	112

ÍNDICE DE TABLAS.

Tabla 1: Resultados de las Pruebas BC 1.	37
Tabla 2: Resultados de las pruebas BC 2.	38
Tabla 3: Resultados de las pruebas BC 1.	38
Tabla 4: Resultados de las pruebas BC 2.	39
Tabla 5: Resultados de las pruebas BC 1.	39
Tabla 6: Tabla 4: Resultados de las pruebas BC 2.	40
Tabla 7: Lista de reserva del producto.	43
Tabla 8: Actores del sistema.	46
Tabla 11: Historia de Usuario Clasificar nuevo caso.	47
Tabla 12: Plan de iteraciones.	48
Tabla 15: Tarjeta CRC ClassificationAlgorithm.	49
Tabla 16: Tarea No.1 de la HU No 1	62
Tabla 19: Tarea No.2 de la HU No.6	63
Tabla 21: Tarea No.1 de la HU No.9	63
Tabla 27: Caso de Prueba de Aceptación Clasificar nuevo caso.	65
Tabla 28: Resultado de la comparación	68
Tabla 29: Tiempo de la Evaluación Empírica.	69
Tabla 30: Tiempos de Evaluación Empírica y los clasificadores del sistema.	70
Tabla 31: Historia de Usuario Importar base de casos.	81
Tabla 32: Historia de Usuario Pre-procesar base de casos.	82
Tabla 33: Historia de Usuario Configurar clasificadores.	82
Tabla 34: Historia de Usuario Entrenar Clasificadores.	83
Tabla 35: Historia de Usuario Evaluar Clasificadores.	84
Tabla 36: Historia de Usuario Exportar clasificadores entrenados.	84
Tabla 37: Historia de Usuario Importas clasificadores entrenados.	85
Tabla 38: Historia de Usuario Visualizar clasificadores entrenados.	85
Tabla 39: Historia de Usuario Crear nuevo caso.	86
Tabla 40: Historia de Usuario Seleccionar clasificadores.	86
Tabla 41: Historia de Usuario Clasificar nuevo caso.	87
Tabla 42: Historia de Usuario Recomendar algoritmos de descubrimiento.	88
Tabla 43: Tarjeta CRC ClassifierManager	89
Tabla 44: Tarjeta CRC Instances.	89
Tabla 45: Tarjeta CRC Instance.	89
Tabla 46: Tarjeta CRC Attribute.	90
Tabla 47: Tarjeta CRC Evaluation.	90
Tabla 48: Tarjeta CRC ClassificationAlgorithm.	90
Tabla 49: Tarjeta CRC AbstractClassifier.	91
Tabla 50: Tarea No.1 de la HU No 1	91
Tabla 51: Tarea No.2 de la HU No.1	92
Tabla 52: Tarea No.1 de la HU No.2.	92
Tabla 53: Tarea No.2 de la HU No.2.	92
Tabla 54: Tarea No.1 de la HU No.3.	93
Tabla 55: Tarea No.2 de la HU No.3.	93
Tabla 56: Tarea No.1 de la HU No.4.	93
Tabla 57: Tarea No.2 de la HU No.4.	94

Tabla 58: Tarea No.1 de la HU No.5	94
Tabla 59: Tarea No.2 de la HU No.5	94
Tabla 60: Tarea No.1 de la HU No.6	95
Tabla 61: Tarea No.2 de la HU No.6	95
Tabla 62: Tarea No.1 de la HU No.7	95
Tabla 63: Tarea No.2 de la HU No.7	96
Tabla 64: Tarea No.1 de la HU No.8	96
Tabla 65: Tarea No.2 de la HU No.8	96
Tabla 66: Tarea No.1 de la HU No.9	97
Tabla 67: Tarea No.2 de la HU No.9	97
Tabla 68: Tarea No.1 de la HU No.10.	97
Tabla 69: Tarea No.2 de la HU No.10.	98
Tabla 70: Tarea No.1 de la HU No.11.	98
Tabla 71: Tarea No.2 de la HU No.11.	98
Tabla 72: Tarea No.1 de la HU No.12.	99
Tabla 73: Tarea No.2 de la HU No.12.	99
Tabla 74: Caso de Prueba de Aceptación Importar base de casos.....	100
Tabla 75: Caso de Prueba de Aceptación Pre-procesar base de casos.	101
Tabla 76: Caso de Prueba de Aceptación Configurar clasificadores.	102
Tabla 77: Caso de Prueba de Aceptación Entrenar clasificadores.	102
Tabla 78: Caso de Prueba de Aceptación Evaluar clasificadores.	103
Tabla 79: Caso de Prueba de Aceptación Exportar clasificadores entrenados.	104
Tabla 80: Caso de Prueba de Aceptación Importar clasificadores entrenados.	104
Tabla 81: Caso de Prueba de Aceptación Visualizar clasificadores entrenados.	105
Tabla 82: Caso de Prueba de Aceptación Crear nuevo caso.	106
Tabla 83: Caso de Prueba de Aceptación Seleccionar clasificadores.	106
Tabla 84: Caso de Prueba de Aceptación Clasificar nuevo caso.	107
Tabla 85: Caso de Prueba de Aceptación Recomendar algoritmos de descubrimiento.	108

INTRODUCCIÓN

Según la IEEE Task Force on Process Mining¹, la Minería de Procesos provee un puente importante entre la Minería de Datos y la modelación y análisis de procesos de negocio. Para esta área del conocimiento, la herramienta más utilizada es el Framework ProM (Claes, Poels 2013). La Minería de Procesos se encuentra dividida en tres áreas o tipos de Minería de Procesos en términos de entradas y salidas que son: **descubrimiento, verificación de conformidad y mejoramiento del modelo**. Siendo el descubrimiento área a la que se le ha prestado mayor atención dentro de la Minería de Procesos. Es donde se utilizan técnicas para tomar un registro de eventos y producir un modelo, que es generalmente un modelo de procesos (red de Petri, BPMN², EPC³, o diagrama de actividad UML⁴) (Arcieri, Stilo, Talamo 2011, p. 1).

Para realizar evaluaciones de rendimiento, identificar anomalías y chequear conformidades entre otros análisis, son necesarios modelos que reflejen con precisión la ejecución real de los procesos descubiertos. Esta necesidad ha impulsado a la creación de varios algoritmos para el descubrimiento de modelos de procesos, enfocados en el orden de ejecución de las actividades (Control de flujo). Un algoritmo de descubrimiento de proceso es una función que mapea un registro de eventos en un modelo de procesos, de tal forma que el modelo “representa” el comportamiento contenido en el registro de eventos (Aalst 2011a).

El descubrimiento de procesos es una tarea compleja y con disímiles aspectos que influyen en la calidad de los modelos descubiertos (Ly, Indiono, Mangler, Rinderle-Ma 2012). El ruido⁵, las tareas duplicadas y las tareas invisibles (ausencia de información) son rasgos del registro de eventos que afectan a los algoritmos de descubrimiento. Se ha determinado además, que la presencia de patrones de control de flujo como el lazo y la elección no libre (van Dongen, De Medeiros, Wen 2009), así como la heterogeneidad de los casos y la baja granularidad de los eventos (Bose 2012), son rasgos del proceso que representan dificultades para los algoritmos de descubrimiento. Además los modelos descubiertos deben poseer un balance entre cuatro criterios de calidad: aptitud, precisión, generalización y simplicidad, cuestión difícil de lograr mayormente porque la precisión y generalización son rasgos opuestos.

¹ Fuerza de Trabajo del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) sobre Minería de Procesos.

² Notación de Modelado de Procesos de Negocios (BPMN).

³ Líneas de proceso gestionadas por eventos (EPC).

⁴ Lenguaje de Modelado Unificado (UML)

⁵ Ruido: Comportamiento raro e infrecuente almacenado en el registro de eventos y que no es representativo del comportamiento típico o común del proceso. (Aalst 2011b)

Hasta el momento no se cuenta con soluciones completas e integrales para las dificultades ya mencionadas, por lo cual la efectividad de los algoritmos de descubrimiento varía según los rasgos del registro de eventos y del proceso asociado. A esto se le suma el hecho de que no todos los algoritmos reaccionan positivamente ante registros de eventos que poseen tareas duplicadas, lo que implica que estos algoritmos mostrarán peores resultados frente a otros.

Un mismo algoritmo de descubrimiento de procesos puede obtener excelentes resultados en un registro de evento dado y sin embargo puede mostrar resultados de baja calidad frente a otros registros de eventos. Por lo que se debe seleccionar para cada registro de eventos, el algoritmo que mejor se ajuste a sus características. Por esta razón es muy frecuente realizar una selección inadecuada de un algoritmo de descubrimiento para un registro de eventos.

Un algoritmo de descubrimiento de proceso que sea inadecuado para un registro de eventos, puede conducir a demoras provocadas por su tiempo de ejecución y ajuste, y a la obtención de un modelo de baja calidad (en algunas situaciones puede no generar ningún modelo). A su vez un modelo de baja calidad puede provocar interpretaciones inadecuadas del proceso minado. Existen algoritmos que al ser aplicados a registros de eventos que contienen eventos muy detallados y procesos poco estructurados, generan modelos de procesos difíciles de comprender, denominados **modelos spaghetti**. (Aalst 2011b).

Con el fin de identificar los algoritmos de descubrimiento que permitan obtener modelos de mejor calidad ante ciertas situaciones, se han desarrollado y aplicado un conjunto de técnicas para la selección de los algoritmos tales como: **Modelos de regresión, Árboles de decisión y Evaluación Empírica**.

La **Evaluación Empírica** es una de las técnicas de la Minería de Procesos empleadas para la selección de algoritmos de descubrimiento. Consiste en probar un conjunto de algoritmos de descubrimiento sobre un mismo registro de eventos y luego evaluar los resultados obtenidos con una serie de métricas de calidad, para seleccionar los algoritmos que hayan obtenido los mejores rendimientos. El rendimiento de estos algoritmos se determina a partir de evaluar la calidad de los modelos construidos por ellos. Para esto se utilizan métricas de calidad que se agrupan en dos métodos fundamentales: **modelo-log** que compara el modelo descubierto con respecto al registro de eventos y **modelo-modelo** que evalúa la paridad entre el modelo descubierto y un modelo de referencia del proceso (De Weerd, De Backer, Vanthienen, Baesens 2011). La ejecución de cada métrica y cada algoritmo tiene un costo computacional asociado que varía en función de sus características.

El uso de esta técnica es poco adecuado para la selección de algoritmos de descubrimiento ya que resulta costosa desde el punto de vista computacional y de tiempo (Wang, Wong, Ding, Guo, Wen 2012). Para ejemplificar lo planteado, en una de las evaluaciones empíricas publicadas que muestra los tiempos necesarios para evaluar 8 métricas sobre 8 registros de eventos reales, los tiempo requeridos para evaluar los algoritmos Genetic Miner, Distributed Genetic Miner y AGNES Miner, oscilaron entre 23 minutos y 22 días. Al tiempo de ejecución debe se le debe adicionar además, el tiempo necesario para el ajuste de los parámetros de configuración de los algoritmos y las métricas a utilizar.

Por estos inconvenientes que influyen en el tiempo requerido para la selección de algoritmos de descubrimiento se propone el siguiente **Problema de investigación**: ¿Cómo disminuir el tiempo requerido para la selección de los algoritmos de descubrimiento de procesos teniendo en cuenta el conocimiento disponible en la Minería de Procesos?

Se propone además como **Objeto de estudio**: Técnicas de Minería de Datos basadas en Aprendizaje Automático (Supervisado), centrando especial atención en la Técnica de Clasificación de Minería de Datos basada en el Aprendizaje Automático orientada a la Minería de Procesos como **Campo de Acción**.

Para darle solución a dicha problemática se propone como **Objetivo general**: Desarrollar un complemento para ProM que permita la clasificación de registros de eventos disminuyendo el tiempo requerido para la selección de algoritmos de descubrimiento de procesos. Desglosado en los siguientes **Objetivos específicos**:

- Definir las características de los registros de eventos que afectan los algoritmos de descubrimiento y las métricas para la evaluación de su rendimiento.
- Valorar la efectividad de clasificadores conocidos frente a una base de conocimiento con información sobre el rendimiento de los algoritmos de descubrimiento frente a registros de eventos con determinadas características.
- Desarrollar un complemento para ProM a partir de los clasificadores que demostraron mayor efectividad en el estudio realizado.
- Validar la propuesta implementada contrastando los resultados obtenidos por los clasificadores implementados en ProM y los clasificadores de WEKA y comparando los

tiempos de la selección por Evaluación Empírica y los tiempos de la evaluación utilizando los clasificadores implementados.

Idea a defender

El desarrollo de un complemento para ProM que permita la clasificación de registros de eventos disminuirá el tiempo requerido para la selección de algoritmos de descubrimiento de procesos.

Resumen de los contenidos de los capítulos.

El presente trabajo está estructurado en 3 capítulos, a continuación se muestra una breve explicación del contenido correspondiente a cada uno de ellos para su mejor entendimiento.

Capítulo 1. FUNDAMENTOS TEÓRICOS

En este capítulo se hace un análisis sobre los conceptos más relevantes relacionados con la Minería de Procesos y la Minería de Datos. Hace énfasis en los algoritmos de descubrimiento y clasificación estudiados en correspondencia como las diferentes técnicas existentes para su evaluación. Se hace un estudio de las diferentes herramientas, tecnologías y lenguajes utilizados, así como el análisis correspondiente para la selección de la metodología que guía el desarrollo del complemento.

Capítulo 2. PRUEBAS A LOS CLASIFICADORES, PLANIFICACIÓN Y DISEÑO

Este capítulo se encuentra dividido en dos momentos importantes: la selección de los cinco algoritmos de clasificación que fueron implementados y la planificación y diseño de la propuesta de solución. Para la selección se tuvo en cuenta el rendimiento mostrado por los algoritmos de clasificación estudiados en tres entornos disponibles en la herramienta Weka. Seguidamente se muestra la planificación y diseño de la propuesta en correspondencia con la metodología XP.

Capítulo 3. IMPLEMENTACIÓN Y PRUEBA

Se presentan los aspectos relacionados con la implementación y pruebas realizadas al sistema. Describe como está implementada la solución, los estándares de codificación utilizados, el desarrollo de las historias de usuarios por iteraciones, ejemplos de código implementado, pruebas realizadas al complemento y los resultados obtenidos. Concluye con la validación de la solución propuesta.

1. FUNDAMENTOS TEÓRICOS.

El presente capítulo aborda los conceptos fundamentales asociados al dominio de la problemática planteada. Describe las técnicas para la selección de algoritmos de descubrimiento de procesos y las categorías en las que se agrupan los algoritmos de clasificación. Definen la metodología de desarrollo de software utilizada así como los lenguajes, tecnologías y herramientas que dieron soporte a la planificación, diseño, implementación y validación de la propuesta de solución.

1.1. Minería de Procesos.

Un concepto fundamental para entender la Minería de Procesos es **Proceso de Negocio**, que para este trabajo se tomará el de Mathias Weske: Un proceso de negocio es una colección de actividades que son realizadas coordinadamente en un ambiente técnico y organizacional. La conjunción de estas actividades logra un objetivo del negocio. Cada proceso de negocio es ejecutado por una simple organización, pero con él pueden interactuar procesos de negocios de otras organizaciones” (Weske 2012).

La **Minería de Procesos** es una disciplina de investigación relativamente joven que se ubica entre la inteligencia computacional y la Minería de Datos por una parte, y la modelación y análisis de procesos por otra. La idea de la Minería de Procesos es **descubrir, monitorear y mejorar** los procesos reales (no los procesos supuestos) a través de la extracción de conocimiento de los registros de eventos ampliamente disponibles en los actuales sistemas (de información). La Minería de Procesos incluye el descubrimiento (automático) de procesos (extraer modelos de procesos a partir de un registros de eventos), la verificación de conformidad (monitorear desviaciones al comparar el modelo y el registro de eventos), la minería de redes sociales organizacionales, la construcción automática de modelos de simulación, la extensión de modelos, la reparación de modelos, la predicción de casos y las recomendaciones basadas en historia. La Minería de Procesos provee un puente importante entre la Minería de Datos y la modelación y análisis de procesos de negocio (Arcieri, Stilo, Talamo 2011, p. 1).

Algoritmos de Descubrimiento:

Los algoritmos de descubrimiento surgen a partir de la necesidad de crear modelos de procesos que reflejen con precisión la ejecución real de los procesos. A finales del 2011 se contaban 26

algoritmos diferentes (De Weerd, De Backer, Vanthienen, Baesens 2011). Los algoritmos de descubrimiento han sido clasificados en las siguientes categorías:

- Basados en abstracciones.
- Basados en heurísticas.
- Basados en búsquedas.
- Basados en lenguajes.
- Basados en descubrimiento de estados (van Dongen, De Medeiros, Wen 2009).

1.2. Técnicas para la selección de los algoritmos de descubrimiento.

Con el fin de identificar los algoritmos de descubrimiento que permiten obtener modelos de mejor calidad ante ciertas situaciones, se han desarrollado y aplicado un conjunto de técnicas para la evaluación de algoritmos. Existen tres técnicas fundamentales que persiguen este objetivo: **Modelos de Regresión, Árboles de decisión y Evaluación Empírica.**

1.2.1. Modelos de Regresión.

El trabajo de Wang (Wang, Wong, Ding, Guo, Wen 2012) es un gran esfuerzo para minimizar el problema de la Evaluación Empírica en la selección de los algoritmos de descubrimiento. El marco propuesto se basa en la selección de los modelos de referencia de alta calidad, construyendo a partir de estos un modelo de regresión, que permite estimar la similitud de otros modelos de procesos sin realizarles una Evaluación Empírica. Este enfoque se compone de una fase de aprendizaje y una fase de recomendación.

Con las características estructurales significativas de los modelos de referencias y los valores de similitud obtenidos después de la Evaluación Empírica de algoritmos, se construye el modelo de regresión durante la fase de aprendizaje. En la fase de la recomendación de los modelos de referencia se extraen las características con el fin de predecir los resultados de similitud con el modelo de regresión. A partir de la similitud estimada se valora el algoritmo ideal para descubrir los procesos asociados con el modelo de referencia que se propone.

Los experimentos realizados utilizando esta solución de recomendación muestran resultados alentadores (Wang, Wong, Ding, Guo, Wen 2012). La evaluación sobre un conjunto de 621 modelos (incluyendo modelos artificiales y reales) ha superado el 90% de precisión en las

recomendaciones. Sin embargo, este enfoque implica algunos requisitos que limitan seriamente su aplicación.

El requisito principal limitante es la necesidad de modelos de referencia para la evaluación y predicción. En varios entornos del mundo real, donde los algoritmos de descubrimiento deben aplicarse, los modelos de proceso no se describen o son inconsistentes y/o incompletos. El enfoque de Weng supone que la ejecución real de los procesos mantiene una estrecha relación con su modelo de referencia. Por lo tanto, en contextos en que las características de los registros reales difieran de los registros generados artificialmente por los modelos de referencia se puede esperar que los resultados sean inexactos. La construcción del modelo de regresión a partir de las características del modelo de referencia descarta problemas como el ruido, la falta de información y la integridad de los registros de eventos, los cuales tienen un impacto significativo en el rendimiento de los algoritmos de descubrimiento.

Por último, pero no menos importante, el marco propuesto ha sido concebido para el uso de las redes de Petri como el lenguaje de modelado. Por lo tanto, sólo es aplicable en los entornos donde se especifican los modelos de uso de las redes de Petri o anotaciones equivalentes. Esta restricción también implica que el marco sólo podía recomendar algoritmos basados en redes de Petri o notaciones equivalentes, con exclusión de los algoritmos, como el Fuzzy Miner (Pérez-Alfonso, Yzquierdo-Herrera, Lazo-Cortés 2014). Como se mencionó antes, este algoritmo es especialmente útil en contextos donde los algoritmos de descubrimiento que generan las redes de Petri son ineficientes.

1.2.2. Árboles de Decisión.

Una alternativa para resolver el problema de selección de algoritmos de descubrimiento es ofrecida por Lakshmanan y Khalaf (Lakshmanan, Khalaf 2012). Los autores proponen construir un árbol de decisión a partir de la comparación de cinco algoritmos de descubrimiento. La comparación establece las potencialidades de los algoritmos para hacer frente a retos como: tareas invisibles y duplicadas, lazos, paralelismos, elección no libre y el ruido.

Sin embargo, no se especifica la forma de identificar la presencia de situaciones difíciles en el proceso a minar. Si bien el reconocimiento de patrones de control de flujo complicados se puede realizar a partir de un modelo de referencia, esto implica los inconvenientes ya discutidos. Por otra parte, la identificación de tareas invisibles, tareas duplicadas, ruido, bucles, paralelismos o elección no libre de un registro de eventos está lejos de ser una tarea trivial. En

general, este estudio ofrece algunos elementos teóricos importantes, pero carece de aplicabilidad práctica directa debido al tipo de información requerida.

1.2.3. Evaluación Empírica.

Para esta técnica el rendimiento de los algoritmos de descubrimiento se determina a partir de la evaluación de la calidad de los modelos que obtienen. Las métricas de calidad definidas se agrupan en dos métodos fundamentales (De Weerd, De Backer, Vanthienen, Baesens 2011). Un método es el que compara el modelo descubierto con respecto al registro de eventos y se denomina modelo-log. El otro método, llamado modelo-modelo, evalúa la paridad entre el modelo descubierto y un modelo de referencia del proceso.

Para la evaluación de algoritmos se ha propuesto un framework⁶ que emplea ambos métodos de evaluación (van der Aalst, van Dongen, Günther, Mans, De Medeiros, Rozinat, Rubin, Song, Verbeek, Weijters 2007). El tiempo necesario para la Evaluación Empírica de algoritmos está determinado por el tiempo de ejecución de cada uno de los algoritmos de descubrimiento a evaluar y el tiempo de ejecución de las métricas a utilizar.

Basada en una parte de esta solución, ha sido publicada otra recientemente, que incluye una fase de optimización de parámetros de los algoritmos de descubrimiento (Ma 2012). Ambas técnicas, al seguir un enfoque de Evaluación Empírica, son poco adecuadas para la selección de algoritmos de descubrimiento ya que resultan costosas desde el punto de vista computacional y de tiempo (Wang, Wong, Ding, Guo, Wen 2012).

1.3. Recomendación como un problema de Clasificación.

Se puede apreciar que existen limitaciones en las soluciones planteadas en el área de Minería de Procesos. Por tanto se trató de buscarle solución a la misma en un área del conocimiento más estudiada y conocida: la Minería de Datos. Esto permitió el uso de otras técnicas y métodos y por consiguiente brindó otro enfoque a la problemática planteada.

Debido a que las técnicas para la selección de algoritmos de descubrimiento antes mencionadas, no dan una solución integral y completa a la problemática planteada se propone que, la recomendación de algoritmos de descubrimiento se exprese en términos de un problema de clasificación. La clasificación es un problema relativo a la construcción de un

⁶ Por su significado en inglés Marco de Trabajo.

procedimiento que se aplica a una secuencia continua de los casos, en la que cada nuevo caso debe ser asignado a una de un conjunto de clases predefinidas sobre la base de atributos o características observadas. Un registro de eventos en el que es necesario recomendar un algoritmo de descubrimiento es considerado como un nuevo caso a ser clasificado. El algoritmo recomendado es la clase predefinida para ser asignada a un registro de eventos en base en las características observadas (Pérez-Alfonso, Yzquierdo-Herrera, Lazo-Cortés 2012). Este tipo de procedimiento de clasificación en el que se conocen las verdaderas clases, ha sido denominado de diversas maneras como: reconocimiento de patrones, discriminación o aprendizaje supervisado (Michie, Spiegelhalter, Taylor 1994).

Modelando la recomendación de algoritmos de descubrimiento como un problema de clasificación se abrió camino para la aplicación de técnicas desarrolladas en esta área del conocimiento.

1.4. Minería de Datos.

Para enfrentarse a un problema de Clasificación es necesario hacerse del conocimiento de esta materia que ayude a su mejor entendimiento, por lo que a continuación se presentan las principales definiciones, conceptos y técnicas que apoyan a su comprensión.

Aprendizaje Automatizado

El aprendizaje automático es la disciplina que estudia cómo construir sistemas computacionales que mejoren automáticamente mediante la experiencia. En otras palabras, se dice que un programa ha “aprendido” a desarrollar la tarea T si después de proporcionarle la experiencia E el sistema es capaz de desempeñarse razonablemente bien cuando nuevas situaciones de la tarea se presentan. Aquí E es generalmente un conjunto de ejemplos de T, y el desempeño es medido usando una métrica de calidad P. Por lo tanto, un problema de aprendizaje bien definido requiere que T, E y P estén bien especificados (Téllez 2005).

Técnicas de Minería de Datos basadas en Aprendizaje Automatizado.

El KDD⁷ es el proceso completo de extracción de información de base de datos que se encarga además de la preparación de los datos y de la interpretación de los resultados obtenidos. Este

⁷Por sus siglas en inglés: Descubrir el conocimiento en bases de datos.

proceso se divide en etapas, siendo una de ellas la **Minería de Datos**, en la que se aplican técnicas que intentan obtener patrones o modelos a partir de los datos recopilados.

Las técnicas de Minería de Datos se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas (Edna, Eléctrica 2006).

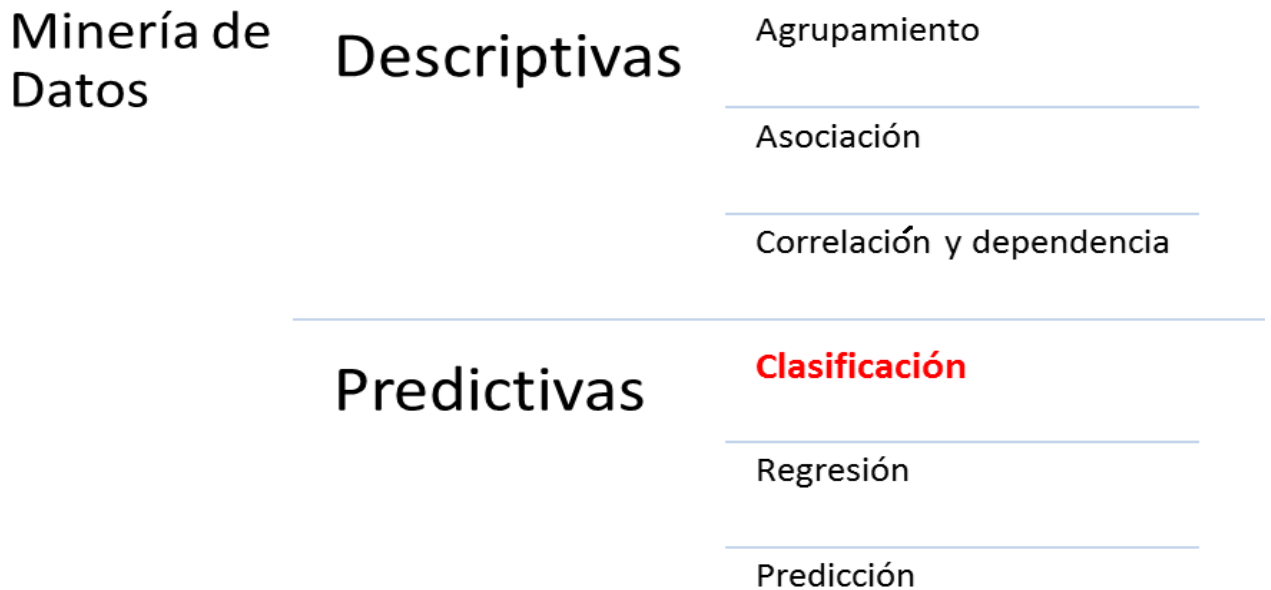


Figura 1: Representación de las categorías de Minería de Datos.

Clasificación.

La clasificación puede ser formalizada como la tarea de aproximar una función objetivo desconocida $\phi = I \times C \rightarrow \{T, F\}$ (que describe cómo instancias del problema deben ser clasificadas de acuerdo a un experto en el dominio) por medio de una función $\Theta = I \times C \rightarrow \{T, F\}$ llamada el clasificador, donde $C = \{C_1, \dots, C_{|C|}\}$ es un conjunto de categorías predefinido, e I es un conjunto de instancias del problema. Comúnmente cada instancia $i_j \in I$ es representada como una lista $A = \langle a_1, a_2, \dots, a_{|A|} \rangle$ de valores característicos, conocidos como atributos, i.e. $i_j = \langle a_{1j}, a_{2j}, \dots, a_{|A|j} \rangle$. Si $\phi = i_j \times c_i \rightarrow T$, entonces i_j es llamado un ejemplo positivo de c_i mientras si $\phi = i_j \times c_i \rightarrow F$ este es llamado un ejemplo negativo de c_i .

Para generar automáticamente el clasificador de c_i es necesario un proceso inductivo, llamado el aprendizaje el cual por observar los atributos de un conjunto de instancias preclasificadas bajo c_i o \bar{c}_i , adquiere los atributos que una instancia no vista debe tener para pertenecer a la categoría.

Por tal motivo, en la construcción del clasificador se requiere la disponibilidad inicial de una colección Ω de ejemplos tales que el valor de $\phi\{i_j, c_i\}$ es conocida para cada $(i_j, c_i) \in \Omega \times C$. A

la colección usualmente se le llama conjunto de entrenamiento (T_r). En resumen, al proceso anterior se le identifica como aprendizaje supervisado debido a la dependencia de T_r (Téllez 2005).

1.5. Algoritmos de Clasificación.

Por el gran número de algoritmos investigados se decidió dividirlos por categorías según las fuentes consultadas (Witten, Frank 2005; Han, Kamber, Pei 2006; Witten, Frank 2005).

1.5.1. Árboles de decisión.

Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica. Donde cada nodo representa un atributo o rasgo y cada hoja representa una solución o clasificación, de tal manera que la decisión final se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas.

Los árboles de decisión son especialmente apropiados para expresar procedimientos médicos, legales, comerciales, estratégicos, matemáticos, lógicos entre otros. Se caracterizan por la sencillez de representación y su forma de operar ante la clasificación de un nuevo caso. Además son de fácil interpretación ya que pueden ser expresados en forma de reglas de decisión (Han, Kamber, Pei 2006).

Entre los algoritmos de este tipo, los más difundidos son el ID3, el C4.5 y el CART, aunque existen otros como el C5.0. De manera general estos algoritmos se caracterizan por su simplicidad conceptual, facilidad uso, rapidez computacional y robustez frente a la falta de información y al ruido. Debido a esto son posiblemente los métodos de aprendizaje más conocidos y más usados en la Minería de Datos (Han, Kamber, Pei 2006).

El desarrollo de un árbol de decisión corresponde a la fase de entrenamiento del modelo y es regulado por un proceso recursivo de heurística natural. Basándose en la filosofía divide y vencerás, se va particionado el esquema de manera inducida de arriba hacia abajo o lo que es lo mismo, desde la raíz a las hojas.

La clasificación de un ejemplo nuevo del que se desconoce su clase se hace con la misma técnica, solamente que en ese caso al atributo clase, cuyo valor se desconoce, se le asigna de acuerdo con la etiqueta de la hoja a la que se accede con ese ejemplo.

Una de las grandes ventajas de los árboles de decisión es que, en su forma más general, las opciones posibles a partir de una determinada condición son excluyentes. Esto permite analizar una situación y siguiendo el árbol de decisión apropiadamente llegar a una sola acción o decisión a tomar (Molina López, García Herrero 2006).

1.5.2. Clasificación Bayesiana.

Los clasificadores bayesianos son clasificadores estadísticos, que pueden predecir tanto las probabilidades del número de miembros de clase, como la probabilidad de que una muestra dada pertenezca a una clase particular (Han, Kamber, Pei 2006).

La clasificación Bayesiana se basa en el teorema de Bayes, y los clasificadores Bayesianos han demostrado una alta exactitud y velocidad cuando se han aplicado a grandes bases de datos. Diferentes estudios comparando los algoritmos de clasificación han determinado que un clasificador Bayesiano sencillo conocido como el clasificador "Naive Bayesiano" es comparable en rendimiento a un árbol de decisión y a clasificadores de redes de neuronas (Chakrabarti, Cox, Frank, Güting, Han, Jiang, Kamber, Lightstone, Nadeau, Neapolitan 2008).

Los clasificadores Naive Bayes suponen que el efecto del valor de un atributo en una clase dada es independiente de los valores de los otros atributos. Este supuesto se llama independencia condicional de clases. Está hecho para simplificar los cálculos necesarios y, en este sentido, es considerado "ingenuo" (naive).

Las Redes bayesianas (Bayes Net) son modelos gráficos, que a diferencia del clasificador Naive bayes, permiten la representación de las dependencias entre los subconjuntos de atributos. Las Redes de creencias bayesianas se pueden también utilizar para la clasificación (Chakrabarti, Cox, Frank, Güting, Han, Jiang, Kamber, Lightstone, Nadeau, Neapolitan 2008).

1.5.3. Clasificación basada en Reglas.

Las técnicas de Inducción de Reglas permiten la generación y contraste de árboles de decisión o reglas y patrones a partir de los datos de entrada. La información de entrada será un conjunto de casos donde se ha asociado una clasificación a un conjunto de variables o atributos. Con esa información estas técnicas obtienen el árbol de decisión o conjunto de reglas que soportan la evaluación o clasificación. En los casos en que la información de entrada posee algún tipo de "ruido" o defecto (insuficientes atributos o datos, atributos irrelevantes o errores u omisiones en los datos) estas técnicas pueden habilitar métodos estadísticos de tipo probabilístico para generar árboles de decisión recortados o podados. También en estos casos pueden identificar los atributos irrelevantes, la falta de atributos discriminantes o detectar "gaps" o huecos de conocimiento. Esta técnica suele llevar asociada una alta interacción con el analista de forma que éste pueda intervenir en cada paso de la construcción de las reglas, bien para aceptarlas, bien para modificarlas.

La inducción de reglas se puede lograr fundamentalmente mediante dos caminos: generando un árbol de decisión y extrayendo de él las reglas, como puede hacer el sistema C4.5 o bien mediante una estrategia de covering⁸, que consiste en tener en cuenta cada vez una clase y buscar las reglas necesarias para cubrir [cover] todos los ejemplos de esa clase; cuando se obtiene una regla se eliminan todos los ejemplos que cubre y se continúa buscando más reglas hasta que no haya más ejemplos de la clase.

Entre los algoritmos de esta categoría se encuentra el 1R (OneR) que se basa en árboles de decisión, el PRISM basado en covering y el PART que es una mezcla de las dos filosofías (Molina López, García Herrero 2006).

1.5.4. Aprendizaje Basado en Ejemplares.

El aprendizaje basado en ejemplares o instancias tiene como principio de funcionamiento, en sus múltiples variantes, el almacenamiento de ejemplos: en unos casos todos los ejemplos de entrenamiento, en otros solo los más representativos, en otros los incorrectamente clasificados cuando se clasifican por primera vez, etc. La clasificación posterior se realiza por medio de una función que mide la proximidad o parecido. Dado un ejemplo para clasificar se le clasifica de

⁸ Por su significado en inglés: Cubrimiento.

acuerdo al ejemplo o ejemplos más próximos. El pilar que riges este método es la proximidad; es decir, la generalización se guía por la proximidad de un ejemplo a otros.

Se han enumerado ventajas e inconvenientes del aprendizaje basado en ejemplares, pero se suele considerar no adecuado para el tratamiento de atributos no numéricos y valores desconocidos. Las mismas medidas de proximidad sobre atributos simbólicos suelen proporcionar resultados muy disímiles en problemas diferentes. A continuación se muestran dos técnicas de aprendizaje basado en ejemplares: el clasificador de los k-vecinos más próximos (k-Nearest-Neighbor Classifiers) y el k-estrella (K^*).

1.5.4.1. KNN: k-Nearest-Neighbor.

El método de los k-vecinos más próximos (KNN, [k-Nearest Neighbor]) está considerado como un buen representante de este tipo de aprendizaje, y es de gran sencillez conceptual. Se suele denominar método porque es el esqueleto de un algoritmo que admite el intercambio de la función de proximidad dando lugar a múltiples variantes. La función de proximidad puede decidir la clasificación de un nuevo ejemplo atendiendo a la clasificación del ejemplo o de la mayoría de los k ejemplos más cercanos.

Esta "cercanía" se define en términos de una métrica de distancia y admite también funciones de proximidad que consideren el peso o coste de los atributos que intervienen, lo que permite, entre otras cosas, eliminar los atributos irrelevantes. (Chakrabarti, Cox, Frank, Güting, Han, Jiang, Kamber, Lightstone, Nadeau, Neapolitan 2008).

1.5.4.2. K-estrella (K^*).

El algoritmo K^* es una técnica de Minería de Datos basada en ejemplares, donde la medida de la distancia que se emplea entre ejemplares se basa en la teoría de la información. Se puede describir como que es la complejidad de transformar un ejemplar en el otro. El cálculo de la complejidad se basa en primer lugar en definir un conjunto de transformaciones $T = \{t_1, t_2, \dots, t_n, \sigma\}$ para pasar de un ejemplo (valor de atributo) a a uno b. La transformación σ es la de parada y es la transformación identidad ($\sigma(a) = a$). El conjunto P es el conjunto de todas las posibles secuencias de transformaciones descritos en T^* que terminan en σ , y $t(a)$ es una de estas secuencias concretas sobre el ejemplo a. Esta secuencia de transformaciones tendrá una probabilidad determinada $p(t)$, definiéndose la función de probabilidad $P^*(b|a)$ como la

probabilidad de pasar del ejemplo a al ejemplo b a través de cualquier secuencia de transformaciones, tal y como se muestra en la ecuación.

$$P^*(b|a) = \sum_{\bar{t} \in P: t(a)=b} p(\bar{t}) \quad \text{Ec. 1}$$

Esta función de probabilidad cumple con las siguientes propiedades:

$$\sum_b P^*(b|a) = 1; \quad 0 \leq P^*(b|a) \leq 1 \quad \text{Ec. 2}$$

La función K* se define entonces en la ecuación:

$$K^*(b|a) = -\log_2 P^*(b|a) \quad \text{Ec. 3}$$

(Téllez 2005)

1.5.5. Redes Neuronales.

Son modelos matemáticos que se ajustan a los datos que se tienen sin necesidad de hacer ninguna suposición a priori (los métodos bayesianos deben suponer alguna distribución en los datos; la de normalidad es la más usada). Muchos de ellos suponen una generalización a métodos estadísticos usados desde hace mucho tiempo.

Se pueden establecer relaciones no lineales entre conjuntos de datos sin necesidad de conocer el tipo de relación de antemano. Como cualquier modelo matemático se pueden analizar y extraer conclusiones cualitativas de ellos. De hecho, por su gran flexibilidad es absolutamente necesario realizar esto. Son de uso habitual en otras áreas de conocimiento por ser modelos no lineales, no paramétricos y con gran robustez al ruido en los datos.

El elemento fundamental de los clasificadores basados en Redes Neuronales es la Neurona Artificial. Esta consta de elementos de entrada y salida que se procesan en la unidad central, así como de los elementos de procesamiento que permitirán a la neurona generalizar y aprender conceptos. La arquitectura básica de la neurona se muestra a continuación.

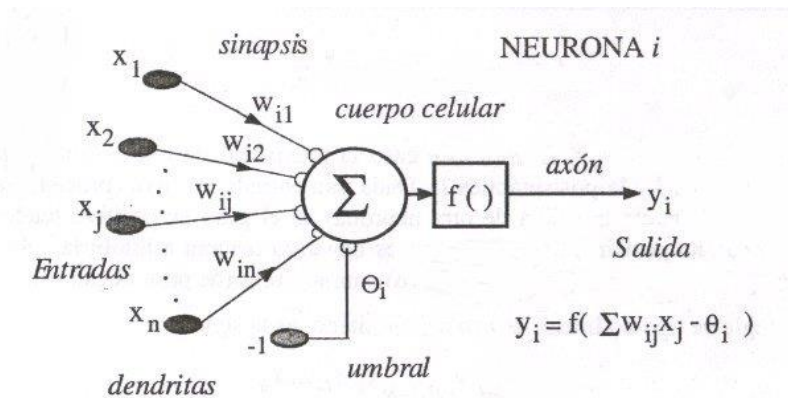


Figura 2: Neurona.

Entre los algoritmos más difundidos soportados esta estructura, están el Perceptrón Simple, Multicapa (MLP) y las Redes de Cuantificación Vectorial (LVQ). Un MLP puede aproximar relaciones no lineales entre datos de entrada y de salida. Además es una de las arquitecturas más utilizadas en la resolución de problemas reales por los siguientes aspectos: por ser un aproximador universal y por su facilidad de uso y aplicabilidad.

Ha sido aplicado con éxito en:

- Reconocimiento de voz.
- Reconocimiento de imágenes.
- Conducción de vehículos.
- Diagnósticos médicos.

El estudio de los algoritmos de clasificación permitió establecer las categorías en las cuales estos se agrupan y el funcionamiento teórico de los mismos. Esto garantizó la selección de aquellos que por su funcionamiento, se adecuaran a las características de la problemática planteada.

1.5.6. Multclasificadores.

Los multclasificadores son conjuntos de clasificadores diferentes que realizan predicciones que se fusionan y se obtiene como resultado la combinación de cada una de ellas. El hecho o la idea de combinación hace que los multclasificadores sean citados en la literatura a través de distintos términos, entre ellos: métodos de ensamble, modelos múltiples (multiple models), sistemas de múltiples clasificadores (multiple classifier systems), combinación de clasificadores

(combining classifiers), integración de clasificadores (integration of classifiers), mezcla de expertos (mixture of experts), comité de decisión (decision committee), comité de expertos (committee of experts), fusión de clasificadores (classifier fusion) y aprendizaje multimodelo (multimodel learning) (Webb 2000).

Se dividen en dos grupos: los métodos de ensamble o ensamblaje y los métodos híbridos, aunque existen otros métodos relacionados que también serán explicados. Los primeros se refieren a aquellos conjuntos de modelos que se combinan para crear un nuevo modelo, empleando para ello la misma técnica de aprendizaje. En el caso de los segundos, son combinaciones de algoritmos de aprendizaje que crean a su vez nuevas técnicas de aprendizaje híbridas (Segrera Francia, García, Navelonga 2006).

Los multclasificadores se distinguen unos de otros atendiendo a diversas características. Entre ellas se puede citar: el número de clasificadores individuales acoplados, el tipo de cada clasificador (redes neuronales, árboles de decisión, vecino más cercano, etc.), las características de los subconjuntos usados por cada clasificador del conjunto, la agregación de las decisiones particulares (voto mayoritario, asignación de pesos, subespacio de mejor comportamiento, funciones de promedio, máximo, mínimo, producto, etc.) y el tamaño y la naturaleza de los conjuntos de datos de entrenamiento para los clasificadores (Skurichina, Duin 1998).

La evaluación y comparación entre los distintos multclasificadores se establece a través de indicadores de rendimiento que incluyen el grado de generalización, aprendizaje, clasificación correcta e incorrecta y el tiempo real de ejecución (Segrera Francia, García, Navelonga 2006).

Los multclasificadores pueden adoptar distintas arquitecturas: secuencial, en serie o vertical, paralela u horizontal e híbrida (mezcla de la arquitectura secuencial con la paralela, con interacción, etc.).

En esta categoría se analizaron los algoritmos MultiClassClassifier, FilteredClassifier y ClassificationViaRegression.

1.6. Metodología de desarrollo de software.

Las metodologías de desarrollo de software son un conjunto de pasos y procedimientos que deben seguirse para llevar a cabo el desarrollo de programas con calidad. Surgen ante la

necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software (Piattini 1996).

Clasificar las metodologías es una tarea bien difícil por la gran cantidad de propuestas y diferencias en el grado de detalle, información disponible y alcance que poseen. A grandes rasgos, considerando su filosofía de desarrollo se pueden agrupar en: metodologías ágiles o ligeras y metodologías pesadas o tradicionales.

Esta investigación se caracteriza por estar más orientada al código que a la documentación, está compuesta por un equipo pequeño de desarrollo y hace especial énfasis en los aspectos humanos del equipo de desarrollo y su relación con el cliente. Por estas características se seleccionó una metodología que por su filosofía fuera ágil, debido a que estas metodologías están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

Para escoger la metodología ágil que guía el desarrollo del complemento, se realizó un estudio de las tres metodologías ágiles que según el ranking de agilidad realizado por el Departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia, son las que destacan con significativa diferencia respecto a CMM9. Estas son las siguientes: ASD10, SCRUM y Extreme Programming (XP) (Letelier, Penadés 2006).

Adaptive Software Development (ASD)

Su impulsor es Jim Highsmith. ASD consiste en un cambio de filosofía en las organizaciones pasando de la transición del modelo Comando-Control al modelo Liderazgo Colaboración. Se caracteriza por ser iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. Su ciclo de vida está compuesto por tres fases importantes: especulación, colaboración y aprendizaje. En la primera especulación se inicia el proyecto y se planifican las características del software, se prefija el camino por donde va a ser guiado el desarrollo del software basado en la teoría de los Sistemas Adaptativos Complejos; por su parte en la fase de colaboración se construye la funcionalidad que se especifica en la fase anteriormente mencionada y finalmente en aprendizaje se revisa su calidad y se entrega al

⁹ Por su siglas en Capability Maturity Model.

¹⁰ Por su siglas en Adaptive Software Development.

cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo. Para su revisión se apoyan en las siguientes categorías:

- Calidad del resultado desde la visión del cliente.
- Calidad del resultado desde la perspectiva técnica.
- El funcionamiento del equipo de desarrollo y las prácticas que este utiliza.
- El status del proyecto.

SCRUM

Esta metodología, de forma similar a XP, recoge las necesidades del usuario en un documento con una prioridad asignada para su implementación. Sigue un conjunto de iteraciones, las cuales son planeadas previamente en reuniones del equipo de trabajo. En estas reuniones, el equipo de trabajo decide cuáles serán los elementos prioritarios a incluir en la iteración en cuestión. El chequeo del trabajo realizado se realiza periódicamente, de esta forma se nutre el proceso de desarrollo con los resultados obtenidos. SCRUM también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable (Ruiz, Almanza, Pons 2011).

XP (Extreme Programming)

Creada por Kent Beck, toma pasos de otras metodologías para realizar un desarrollo más agradable y sencillo. La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código. Se plantea como una metodología a desarrollar en proyectos de riesgos.

Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los involucrados, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Propone un número reducido de roles, dentro de los cuales el cliente es uno de ellos. A lo largo del proceso de desarrollo se producen pequeñas partes del sistema que se han ido implementando. El código generado es compartido para todo el equipo, cualquier miembro puede hacer cambios, y es modificado a lo largo de las iteraciones. Una particularidad de esta metodología es que propone la programación en pares, consiste en que

dos desarrolladores participen en un proyecto en una misma estación de trabajo (Ruiz, Almanza, Pons 2011). El ciclo de vida ideal de XP consiste en cuatro fases: Planificación, Diseño, Codificación y Pruebas.

Justificación de la metodología propuesta

Para guiar el desarrollo de esta aplicación se utilizó la metodología ágil XP porque se centra básicamente en la implementación de las soluciones y no en el soporte documental. Además da respuesta y posible solución a cambios repentinos en las funcionales del sistema y concluye un proyecto lo más rápido posible. Logra que cada miembro del equipo incluyendo el cliente de desarrollo esté listo para enfrentar cualquier cambio en el software, promoviendo el trabajo en equipo y preocupándose por el aprendizaje de los desarrolladores. Esta metodología se basa en la realimentación continua entre el cliente y el equipo de desarrollo y tiene como objetivo lograr su satisfacción.

Es idónea para el desarrollo del complemento para ProM ya que el alcance de la implementación no es muy amplio, no existe la exigencia de generar mucha documentación y el equipo de desarrollo es de dos personas propiciando la programación en pares.

1.7. Lenguajes de modelado y desarrollo.

UML.

El Lenguaje Unificado de Modelado (UML) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan (Larman 1999). Existen muchas notaciones y métodos usados para el diseño orientado a objetos, ahora el personal sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

Los objetivos de UML son muchos pero se centra en expresar de forma gráfica el sistema que se desee implementar de una forma entendible y logra especificar cada una de sus características. A partir de los modelos especificados se logran construir los sistemas diseñados

y el diseño se reutilizaría como parte de la documentación del producto (Booch, Rumbaugh, Jacobson, Martínez, Molina 1999).

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada el Proceso Unificado de Desarrollo (I. Jacobson, G. Booch, J. Rumbaugh, 2000).

Java.

Es un lenguaje de programación orientado a objetos, que permite a los programadores realizar aplicaciones de múltiples tipos, ya sean de escritorio o web. Se caracteriza por ser un lenguaje simple, robusto y poderoso que se torna fácil de aprender, debido a que elimina sentencias de bajo nivel además del Recolector de Basura, haciendo transparente para los programadores el manejo de la memoria. Se destaca por ser un lenguaje de código abierto y multiplataforma por lo cual ha logrado una gran expansión por todo el mundo. En la actualidad incluye un gran número de librerías para múltiples trabajos (Naughton, Schildt 1997).

1.8. Herramientas

Weka 3.7.10.

En la actualidad existe una gran cantidad de herramientas tanto libres como privadas para el desarrollo de proyectos de Minería de Datos. A continuación se muestran algunos de las herramientas para Minería de Datos libre más populares que actualmente se encuentran disponibles para el usuario.

- Orange
- Yale: RapidMiner
- Weka, Universidad de Waikato, Nueva Zelanda.
- JHelpWork
- Knime

(Corral Herrerías 2013).

Se realizó un estudio comparativo de las herramientas RapidMiner y Weka por ser de las herramientas libre más utilizadas nivel internacional para Minería de Datos (Molina López, García Herrero 2006). De estas herramientas, la seleccionada para hacer uso en esta investigación es Weka, dado que está implementada en código abierto y sus algoritmos pueden ser llamados desde un código de java propio utilizando la API de Weka. Permite realizar manipulaciones sobre los datos aplicando filtros para la normalización y selección de atributos, conjuntamente se pueden importar datos en varios formatos. A continuación se describe esta herramienta.

Acrónimo de Waikato Environment for Knowledge Analysis (Entorno para el Análisis del Conocimiento de Waikato), es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario. Para ello únicamente se requiere que los datos a analizar se almacenen con un cierto formato, conocido como ARFF(Attribute-Relation File Format) (Hall, Frank, Holmes, Pfahringer, Reutemann, Witten 2009).

WEKA se distribuye como software de libre distribución desarrollado en Java. Está constituido por una serie de paquetes de código abierto con diferentes técnicas de pre-procesado, clasificación, agrupamiento, asociación y visualización. Estos paquetes pueden ser integrados en cualquier proyecto de análisis de datos, e incluso pueden extenderse con contribuciones de los usuarios que desarrollen nuevos algoritmos. Con el objetivo de facilitar su uso por un mayor número de usuarios, WEKA además incluye una interfaz gráfica de usuario para acceder y configurar las diferentes herramientas integradas (Molina López, García Herrero 2006).

WEKA soporta las extensiones como ARFF, CSV, C4.5 y BINARY. Los datos pueden ser leídos desde URL o pueden estar almacenados en una base de datos SQL y ser leídos por la aplicación. La herramienta WEKA tiene funcionalidades de limpiado (Filtros) que permiten hacer limpieza de los datos y tener información más clara de lo que se está buscando. Posee métodos de: discretización, normalización, muestreo, selección de atributos, transformación y combinación de atributos.

ProM 6.2.

ProM es un marco extensible que es compatible con una amplia variedad de técnicas de proceso de minería en forma de complementos. Es independiente de la plataforma ya que es

implementado en Java y puede ser descargado sin ningún coste. El framework ProM está publicado bajo la licencia de código abierto Common Public License (CPL).

En la actualidad, existen más de 120 paquetes conteniendo alrededor de 500 complementos disponibles que son compatibles con la importación (y la conversión) de varios lenguajes de modelado de los procesos. Presenta una gran variedad de filtros de registro, que son una herramienta valiosa para la limpieza de registros de los artefactos no deseados, o sin importancia (ProM 6 2013).

NetBeans 7.3.

Es un entorno de desarrollo integrado (IDE) para múltiples plataformas, que permite crear aplicaciones de escritorio, aplicaciones web y aplicaciones móviles utilizando las últimas tecnologías para los desarrolladores de software de Java. Posibilita la implementación de software utilizando otros lenguajes de programación. Es una aplicación de código abierto bajo licencias libres por lo que tiene una amplia comunidad de desarrollo en constante crecimiento (Boudreau, Glick, Greene, Spurlin, Woehr 2002).

Visual Paradigm 8.0.

Herramienta de software libre para el modelado visual UML que ayuda a la planificación, análisis y diseño, generación de código fuente y documentación de programas informáticos. Además permite representar todo tipo de diagramas en el ciclo de vida del desarrollo de software, también soporta estándares como Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI, entre otros. Está disponible en múltiples plataformas y bajo licencia gratuita y comercial (Paradigm 2010).

JUnit

JUnit es un framework creado por Erich Gamma y Kent Beck, que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente (Gamma, Beck 1999).

1.9. Conclusiones Parciales.

Del presente capítulo se puede concluir que:

- La investigación realizada permitió el análisis de las principales características que afectan el rendimiento de los algoritmos de descubrimiento.
- La recomendación de algoritmos de descubrimiento puede modelarse como un problema de clasificación y permitir el uso de otras técnicas de Minería de Datos.
- El análisis de las características de la investigación demostró que XP es la metodología adecuada para el desarrollo del complemento.

2. PRUEBAS A LOS CLASIFICADORES, PLANIFICACIÓN Y DISEÑO

Introducción.

Este capítulo tiene un enfoque práctico y funcional, para ello se obvian los detalles técnicos y específicos de los diferentes algoritmos que se presentaron en el capítulo anterior y se centra en su aplicación, configuración y análisis dentro de la herramienta Weka. Se evalúan los algoritmos de clasificación y luego se seleccionan los cinco algoritmos con mejores rendimientos. Se muestran las ventajas de la propuesta de solución así como todos los artefactos requeridos para la planificación y el diseño del complemento definidos por la metodología XP.

2.1. Preparación y análisis de los datos.

Por el gran número de algoritmos de clasificación existentes no se pretendió realizar la implementación de todos en el complemento para ProM, por lo que fue necesario hacer una selección de los mismos. Para realizar esta selección se propuso realizar un conjunto de pruebas empíricas de rendimiento mediante el uso de la herramienta Weka. A continuación se describe este proceso.

Para evaluar el rendimiento de los algoritmos de clasificación en entornos diferentes se consideró su evacuación en dos bases de casos diferentes. La primera (en lo adelante BC1) corresponde a la transformación de los datos del artículo “Uncovering the Relationship between Event Log Characteristics and Process Discovery Techniques” (vanden Broucke, Delvaux, Freitas, Rogova, Vanthienen, Baesens 2014), obtenidos gracias a uno de sus autores Vanden Broucke. Esta base de casos fue importada desde el fichero “Base de Casos.csv” correspondientes a dicha transformación. Este fichero contiene muestras correspondientes a 377 tuplas concernientes a la aplicación de 13 algoritmos de descubrimiento sobre un mismo registro de eventos y su evaluación mediante 22 métricas de calidad.

Los algoritmos y métricas registradas en la base de casos son las siguientes:

Algoritmos de descubrimiento	Métricas de calidad
Alpha	Advanced Behavioural Appropriateness #1
AlphaP	Advanced Structural Appropriateness #1
AlphaPP	Alignment Based Fitness #1
DWS	Alignment Based Precision #1

Causal	Alignment Based Probabilistic Generalization #1
Gen	Average Node Arc Degree #1
DTGen	Behavioral Precision #1
Agnes	Behavioral Profile Conformance #1
Heur	Best Align Precision #1
PTM	Count Arcs #1
Transition	Count Cut Vertices #1
Region	Count Nodes #1
ILP	Count Places #1
	Count Transitions #1
	ETC Precision #1
	Fitness #1
	Negative Event Generalization #1
	Negative Event Precision #1
	Negative Event Recall #1
	One Align Precision #1
	Proper Completion #1
	Weighed P/T Average Arc Degree #1

Cada tupla de BC1 representa un caso individual. Un caso esta descrito por la siguiente estructura: rasgos del proceso (ruido, tareas duplicadas, tareas invisibles), características del registro de eventos (lazos, elección no libre), métricas de calidad evaluadas y algoritmo de descubrimiento evaluado.

Se decidió evaluar también a los clasificadores en otra base de casos con otras características (en lo adelante BC2), para medir el desempeño de los clasificadores en entornos diferentes. Dicha base de datos está compuesta por un total de 795 tuplas, que describen la aplicación de 4 algoritmos de descubrimiento sobre 74 registros de eventos y su evaluación mediante 4 métricas de calidad.

Algoritmos de descubrimiento	Métricas de calidad
• Alpha	• Simplicidad
• Heuristic	• Fidness
• Inductive	• Precisión
• ILP	• Generalización

Cada tupla de BC2 representa: un registro de eventos descrito por sus características (Trazas, AND, XOR, LOOP, RUIDO, INTERVALOS, INVISIBLES, TOTAL), el valor de las métricas de calidad (Simplicidad, Fidness, Precisión y Generalización) y el algoritmo de descubrimiento correspondiente.

2.2. Configuración y Métodos de Evaluación.

Para evaluar los resultados de los algoritmos de clasificación o clasificadores, se definieron un conjunto de variables como: tiempo de construcción del modelo (TE), instancias correctamente clasificadas (ICC), instancias incorrectamente clasificadas (IIC) y error cuadrático medio (ECM), que permitieron hacer una evaluación de la calidad del algoritmo y realizar comparaciones entre ellos.

La evaluación sigue un enfoque empírico, o sea, una vez probados todos ellos, se selecciona el algoritmo que mayor cantidad de instancias correctamente clasificadas tuvo y por consiguiente menor error cuadrático medio. También se tuvo en cuenta el tiempo de ejecución en caso de resultados semejantes con los parámetros anteriores.

2.2.1. Modos de evaluación del clasificador.

La evaluación del resultado de aplicar el algoritmo de clasificación se efectúa comparando que la clase predicha corresponda con la clase real de las instancias. Esta evaluación puede realizarse de diferentes modos:

Use training set: esta opción evalúa el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo para determinar el error, que en este caso se denomina "error de re-sustitución". Por tanto, esta opción puede proporcionar una estimación demasiado optimista del comportamiento del clasificador, al evaluarlo sobre el mismo conjunto sobre el que se hizo el modelo.

Supplied test set: evaluación sobre conjunto independiente. Esta opción permite cargar un conjunto nuevo de datos. Sobre cada dato se realizará una predicción de clase para contar los errores.

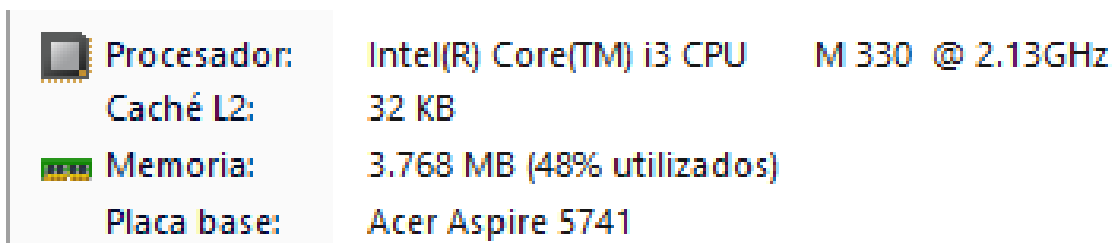
Cross-validation: evaluación con validación cruzada. Esta opción es la más elaborada y costosa. Se realizan tantas evaluaciones como se indica en el parámetro Folds. Se dividen las

instancias en tantas carpetas como indica este parámetro y en cada evaluación se toman las instancias de cada carpeta como datos de test, y el resto como datos de entrenamiento para construir el modelo. Los errores calculados son el promedio de todas las ejecuciones.

Percentage split: esta opción divide los datos en dos grupos, de acuerdo con el porcentaje indicado (%). El valor indicado es el porcentaje de instancias para construir el modelo, que a continuación es evaluado sobre las que se han dejado aparte. Cuando el número de instancias es suficientemente elevado, esta opción es suficiente para estimar con precisión las prestaciones del clasificador en el dominio (Bouckaert, Frank, Hall, Kirkby, Reutemann, Seewald, Scuse 2013).

Para este trabajo se evaluaron los clasificadores en tres de los cuatro entornos mencionados: Use training set, Cross-validation y Percentage Split.

Todos los algoritmos de clasificación fueron evaluados en una misma estación de trabajo con las siguientes características. Esto permitió que todos los algoritmos se ejecutaran en igualdad de recursos.



Procesador:	Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz
Caché L2:	32 KB
Memoria:	3.768 MB (48% utilizados)
Placa base:	Acer Aspire 5741

Figura 3: Características de la Estación de Trabajo.

2.3. Resumen de los resultados de las pruebas.

A continuación se muestra un resumen de las evaluaciones realizadas en ambas bases de casos por entornos de evaluación de Weka.

2.3.1. Entorno: Cross-validation (10 folds).

Tabla 1: Resultados de las Pruebas BC 1.

Algoritmo	TE	ICC	IIC	ECM	Configuraciones
Simple Logic	2,09	253	124	0,1792	Por defecto
ClassificationViaRegression	1,06	241	136	0,1901	Por defecto
MultiClassClassifier	3,18	238	139	0,2511	Clasificador: Simple Logic. Método 1 contra 1
Bayes Net	0,19	235	142	0,2	Por defecto
MLP	12,67	227	150	0,2112	Por defecto
FilteredClassifier	0,03	221	156	0,2015	Por defecto
Naive Bayes	0,02	217	160	0,23	Por defecto
PART	0,39	213	164	0,1969	Por defecto
J48	0,05	211	166	0,1938	Por defecto
OneR	0,01	137	240	0,313	Por defecto
K*	0	40	337	0	Global Blend: 100
IBK	0	39	338	0,3575	Por defecto
Logic	5,18	0	377	0,0359	Por defecto

Tabla 2: Resultados de las pruebas BC 2.

Algoritmo	TE	ICC	IIC	ECM	Configuraciones
MultiClassClassifier	1,04	795	0	0,3118	Clasificador: Simple Logic. Método 1 contra 1
MLP	2,86	794	1	0,0218	Por defecto
Simple Logic	1,3	794	1	0,0558	Por defecto
Bayes Net	0,04	793	2	0,0441	Por defecto
Logic	0,36	793	2	0,0359	Por defecto
OneR	0,01	793	2	0,0355	Por defecto
FilteredClassifier	0,08	792	3	0,0434	Por defecto
ClassificationViaRegression	0,1	792	3	0,0459	Por defecto
PART	0,01	792	3	0,0434	Por defecto
J48	0,01	792	3	0,0434	Por defecto
K*	0	791	4	0,1535	Global Blend: 100
Naive Bayes	0	709	86	0,2	Por defecto
IBK	0	502	293	0,4296	Por defecto

2.3.2. Entorno: Use training set.

Tabla 3: Resultados de las pruebas BC 1.

Algoritmo	TE	ICC	IIC	ECM	Configuraciones
Logic	4,24	324	53	0,1046	Por defecto
MLP	12,96	316	61	0,1232	Por defecto
ClassificationViaRegression	0,54	311	66	0,1554	Por defecto
Simple Logic	0	300	77	0,1464	Por defecto
MultiClassClassifier	0,15	294	83	0,249	Clasificador: Simple Logic.

					Método 1 contra 1
Bayes Net	0,02	276	101	0,1675	Por defecto
FilteredClassifier	0,03	269	108	0,168	Por defecto
J48	0,01	268	109	0,1626	Por defecto
PART	0,14	266	111	0,1608	Por defecto
Naive Bayes	0,01	244	133	0,211	Por defecto
IBK	0,19	211	166	0,2408	Por defecto
OneR	0	165	212	0,4941	Por defecto
K*	1,18	128	249	0,2661	Global Blend: 100

Tabla 4: Resultados de las pruebas BC 2.

Algoritmo	TE	ICC	IIC	ECM	Configuraciones
Logic	0,01	795	0	0	Por defecto
OneR	0	795	0	0	Por defecto
FilteredClassifier	0	795	0	0	Por defecto
PART	0	795	0	0	Por defecto
J48	0	795	0	0	Por defecto
MLP	2,67	795	0	0,005	Por defecto
Bayes Net	0,02	795	0	0,0116	Por defecto
ClassificationViaRegression	0,01	795	0	0,0165	Por defecto
Simple Logic	0,79	795	0	0,0476	Por defecto
MultiClassClassifier	0,02	795	0	0,3118	Clasificador: Simple Logic. Método 1 contra 1
Naive Bayes	0,02	724	71	0,1912	Por defecto
K*	2,59	573	222	0,4325	Global Blend: 100
IBK	0	549	246	0,3924	Por defecto

2.3.3. Entorno: Percentage Split (66%).

Tabla 5: Resultados de las pruebas BC 1.

Algoritmo	TE	ICC	IIC	ECM	Configuraciones
MultiClassClassifier	3,9	82	46	0,2511	Clasificador: Simple Logic. Método 1 contra 1
Simple Logic	2,28	79	49	0,198	Por defecto
ClassificationViaRegression	0,51	76	52	0,2	Por defecto
Naive Bayes	0,03	75	53	0,229	Por defecto
Bayes Net	0,02	74	54	0,209	Por defecto
MLP	12,62	73	55	0,23	Por defecto
PART	0,14	72	56	0,2055	Por defecto
Logic	4,3	68	60	0,2583	Por defecto
FilteredClassifier	0,02	67	61	0,2107	Por defecto

J48	0	67	61	0,2116	Por defecto
OneR	0	46	82	0,3139	Por defecto
IBK	0,02	8	120	0,3642	Por defecto
K*	0,45	5	123	0,2667	Global Blend: 100

Tabla 6: Tabla 4: Resultados de las pruebas BC 2.

Algoritmo	TE	ICC	IIC	ECM	Configuraciones
OneR	0	270	0	0	Por defecto
FilteredClassifier	0	270	0	0	Por defecto
Bayes Net	0	270	0	0,0155	Por defecto
MLP	2,67	270	0	0,0159	Por defecto
ClassificationViaRegression	0	270	0	0,0241	Por defecto
Simple Logic	0	270	0	0,0622	Por defecto
MultiClassClassifier	2,67	270	0	0,3118	Clasificador: Simple Logic. Método 1 contra 1
PART	0	269	1	0,043	Por defecto
J48	0,01	269	1	0,0434	Por defecto
Logic	0	266	4	0,0855	Por defecto
Naive Bayes	0,01	245	25	0,204	Por defecto
IBK	0,04	164	106	0,4414	Por defecto
K*	0,73	62	208	0,443	Global Blend: 100

2.4. Conclusión de las pruebas a los clasificadores.

Una vez concluida las pruebas se percibieron los siguientes aspectos

- Los resultados obtenidos en el **Use training set** fueron los más favorables para ambas bases de casos, debido a que los modelos construidos por los clasificadores fueron evaluados en el mismo conjunto de datos por los que fueron creados.
- En el entorno **Use training set** los algoritmos que más resaltaron para ambas bases de casos fueron: Logic, MLP, ClassificationViaRegression, Simple Logic, MultiClassClassifier, Bayes Net, FilteredClassifier, J48 y PART.
- Para **Cross-validation**: Simple Logic, ClassificationViaRegression, MultiClassClassifier, Bayes Net, MLP, FilteredClassifier, Naive Bayes, PART y J48 fueron los de mejor desempeño.

- En **Percentage split** los más significativos fueron: MultiClassClassifier, Simple Logic, ClassificationViaRegression, Bayes Net, MLP, PART y FilteredClassifier, llegando a un consenso entre los resultados de los clasificadores en ambas bases de datos.

Del resultado de la ejecución de las pruebas se seleccionaron los siguientes algoritmos para su implementación en el complemento para ProM, por considerarse los de mejores resultados:

- Perceptrón Multicapa.
- PART
- J48
- Simple Logic
- ClassificationViaRegression
- MultiClassClassifier
- FilteredClassifier

2.5. Ventajas del uso del complemento para la clasificación.

Como ya se ha mencionado, las herramientas fundamentales para este trabajo son ProM y Weka. Es posible establecer una integración entre dichas herramientas debido a que ambas se construyeron bajo el mismo lenguaje (Java) y Weka puede ser integrado a otras plataformas. Se cuenta con una librería de Weka en ProM que permite entre otras ventajas el uso de los clasificadores seleccionados sin necesidad de su implementación.

Es válido mencionar las ventajas y posibilidades que (una vez desarrollado) brinda el complemento respecto al uso de Weka y a otras herramientas de iguales propósitos.

- Weka posee un gran conjunto de vistas y entornos que sirven para configurar no solo a los clasificadores, sino que también a varios módulos de visualización, agrupamiento, selección de atributos y pre-procesamiento. El desconocimiento de esta herramienta no es una limitación para el uso de la propuesta de solución ya que no es necesario el uso de estos módulos.

- Los algoritmos de clasificación muestran mejores rendimientos cuando se le realizan ciertas transformaciones a la base de casos con los que entrenan. Normalizar y/o Discretizar son ejemplos filtros que hacen que varios algoritmos se adapten a la base de casos y mejoren su rendimiento. Para los usuarios a los que está destinada esta investigación, que por lo general están más relacionados con la Minería de Procesos que con la Minería de Datos, esta situación representa un problema, ya que muchos desconocen los filtros y configuraciones necesarias para realizar dichas transformaciones. La solución propuesta previó este inconveniente y realiza todas estas transformaciones de forma automática.
- Aunque Weka permite importar una base de casos y entrenar a los clasificadores, no permite clasificar un nuevo caso de forma visual. Solo mediante línea de comandos (Simple CLI) o haciendo uso de la API se puede acceder a esta funcionalidad. Por las características que presenta el usuario final de la aplicación, esto representa un obstáculo importante. Por esta razón al clasificar un nuevo caso, la aplicación muestra una interfaz gráfica para conformar el nuevo caso y posteriormente recomendarle un algoritmo de descubrimiento.
- Implementar la propuesta mediante un complemento para ProM, permite la relación entre otros complementos del framework que aportan información referente a las características del proceso, del registro de eventos y de las métricas de calidad.
- La popularidad de ProM posibilita que las potencialidades del complemento lleguen a un mayor número de usuarios.
- El complemento es capaz de recomendar un algoritmo de descubrimiento por cada algoritmo de clasificación implementado, o sea, brinda más de una alternativa en el momento de la selección.
- El sistema está constituido por dos complementos principales, uno de ellos se encarga del entrenamiento de los clasificadores y el otro de la clasificación de los registros de eventos. Los tiempos para la clasificación de los registros de eventos no son relevantes debido a que este proceso se realiza en instantes de tiempo menores que 1 segundo (para la base de casos empleada). El tiempo significativo es el empleado para el entrenamiento de los clasificadores. Este aspecto es contrarrestado debido a que los clasificadores entrenados pueden ser reutilizados para clasificar nuevos casos sin

necesidad de volver a entrenarlos, lo que permite la disminución del tiempo requerido para la recomendación.

2.6. Propuesta de sistema a desarrollar.

Específicamente se propone un complemento para ProM que permite la clasificación de registros de eventos a partir de una base de conocimientos con información sobre el rendimiento de algoritmos de descubrimiento frente a registros de eventos con determinadas características, disminuyendo el tiempo requerido para la selección de algoritmos de descubrimiento de procesos.

Lista de reserva del producto.

Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

Tabla 7: Lista de reserva del producto.

Item	Descripción	Estimación	Estimado por
Muy alta			
1	Entrenar clasificadores	1	Programador
2	Clasificar nuevo caso.	1	Programador
3	Recomendar algoritmos de descubrimiento.	1	Programador
Alta			
1	Configurar clasificadores.	1	Analista.
2	Evaluar clasificadores.	0.5	Programador
3	Crear nuevo caso.	1	Analista.
4	Importar base de casos.	0.5	Analista.
Media			
1	Pre-procesar base de casos.	0.5	Analista.

2	Exportar clasificadores entrenados.	0.5	Programador
3	Importar clasificadores entrenados.	0.4	Programador
Baja			
1	Seleccionar clasificadores	0.3	Programador
2	Visualizar clasificadores entrenados.	0.3	Programador

2.7. Modelo conceptual.

Con la construcción del modelo conceptual se logra una mejor comprensión del dominio del problema. Un modelo conceptual no es más que una representación visual de los conceptos u objetos del mundo real que son significativos para el problema o el área que se analiza, representando las clases conceptuales, no los componentes de software. Puede verse como un modelo que comunica a los interesados, cuáles son los términos importantes y cómo se relacionan entre sí (Larman 1999).



Figura 4: Modelo conceptual.

Base de Conocimiento: Hace referencia a la base de datos con la información almacenada sobre el rendimiento de los algoritmos de descubrimiento frente a registros de eventos con determinadas características.

Clasificador: Responsable de la clasificación de registros de eventos.

Administrador de Clasificación: Encargado de gestionar las operaciones relacionadas con los clasificadores mediante el uso de la base de conocimiento.

Registro de Eventos: Hace referencia al nuevo registro de eventos al que se le pretende recomendar un algoritmo de descubrimiento.

Algoritmo de Descubrimiento: Es el algoritmo de descubrimiento que es recomendado al nuevo registro de eventos.

2.8. Requisitos del software.

La ingeniería de requisitos es la encargada de recoger todos los detalles funcionales y no funcionales obtenidos de un extenso proceso de entendimiento con el cliente. La información recopilada debe traducirse de manera sencilla y bien entendible para que el desarrollo del producto sea parejo a las necesidades del cliente. Los requisitos funcionales definen el comportamiento interno de la aplicación, describen en detalle cada funcionalidad y los no funcionales complementan los funcionales detallando el ambiente a desarrollar o detalles técnicos.

2.9. Requisitos Funcionales.

- RF 1.** Importar Base de casos.
- RF 2.** Pre-procesar base de casos.
- RF 3.** Configurar clasificadores.
- RF 4.** Entrenar clasificadores.
- RF 5.** Evaluar clasificadores.
- RF 6.** Exportar clasificadores entrenados.
- RF 7.** Importar clasificadores entrenados.
- RF 8.** Visualizar clasificadores entrenados.
- RF 9.** Crear nuevo caso.
- RF 10.** Seleccionar clasificadores
- RF 11.** Clasificar nuevo caso.
- RF 12.** Recomendar algoritmos de descubrimiento.

2.10. Requisitos no funcionales.

Usabilidad.

Facilidad de uso por parte de los usuarios: el sistema debe presentar una interfaz que permita una fácil interacción con el usuario. Además la interfaz debe permitir un manejo cómodo que posibilite a los usuarios sin experiencia en la Minería de Datos una rápida adaptación al mismo.

Escalabilidad.

El sistema debe ser capaz de permitir la integración con otros complementos y permitir la adición nuevas funcionalidades a los complementos implementados.

Software.

Para ejecutar el programa se debe contar con una PC con la Máquina Virtual de Java instalada.

Requerimientos del diseño e implementación:

Como condiciones específicas para el uso de herramientas y lenguajes en las fases de construcción del software, se definieron estos requerimientos.

- Debe implementarse en el lenguaje Java.
- Debe implementarse sobre la versión 6.2 o superior del framework ProM.
- Para la implementación del complemento debe utilizarse el IDE NetBeans.

2.11. Descripción de los actores del sistema a automatizar.

Tabla 8: Actores del sistema.

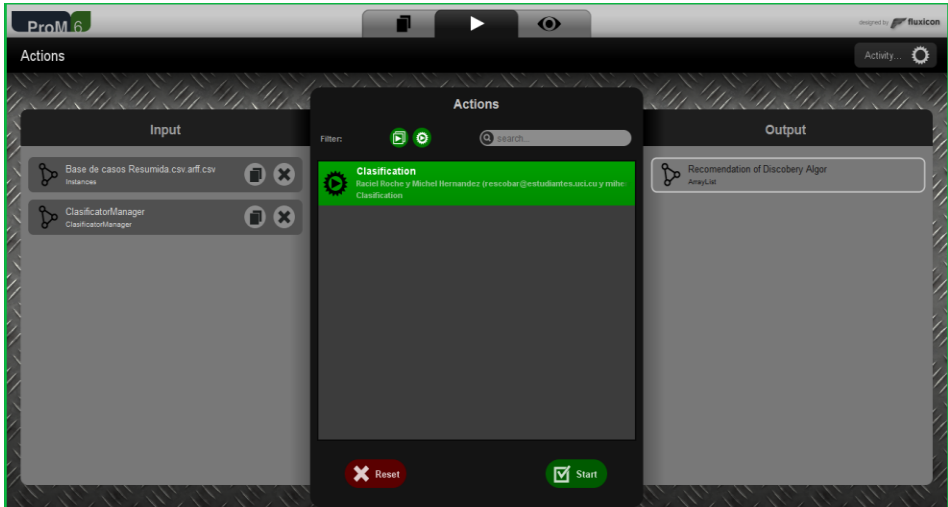
Nombre del Actor	Descripción
Usuario	Puede configurar todos los datos necesarios para la gestión de los clasificadores y la recomendación de algoritmos de descubrimiento.

2.12. Descripción de las historias de usuarios (HU).

Las historias de usuario permiten administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Permiten responder rápidamente a los requisitos cambiantes. Encierran en ellas las funcionalidades que se van a implementar en el sistema (Escribano 2006).

A continuación se muestran las historias de los usuarios más importantes.

Tabla 9: Historia de Usuario Clasificar nuevo caso.

Historia de Usuario	
Número: 11	Nombre Historia de Usuario: Clasificar nuevo caso.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Michel Hernández Pajaro.	Iteración Asignada: 3ra Iteración.
Prioridad en Negocio: Muy Alta	Puntos Estimados: 1 semanas.
Riesgo en Desarrollo: Muy Alta	Puntos Reales: 1 semana.
Descripción. El sistema debe permitirle al usuario clasificar nuevo caso a partir de los clasificadores entrenados.	
Observaciones: La historia de usuario comienza el usuario selecciona la opción clasificar y termina cuando es clasificado el nuevo caso.	
Prototipo de Interfaz:	
	

El resto de las historias de usuarios se encuentran en el Anexo 1.

2.13. Plan de Iteraciones

Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteraciones son las historias de usuario. Una vez identificadas las historias de usuario, se establecieron tres iteraciones para el desarrollo de la aplicación. En la selección de las historias de usuario a implementar se tuvieron en cuenta la prioridad que estas presentan y la relación en cuanto a funcionalidad entre las mismas.

Tabla 10: Plan de iteraciones.

Iteración	Numero de HU	HU	Duración estimada
1	1	Importar base de casos.	3
	2	Pre-procesar base de casos.	
	3	Configurar clasificadores.	
	4	Entrenar clasificadores.	
2	5	Evaluar clasificadores.	1.7
	6	Exportar clasificadores entrenados.	
	7	Importar clasificadores entrenados.	
	8	Visualizar clasificadores entrenados.	
3	9	Crear nuevo caso.	3.3
	10	Seleccionar clasificadores	
	11	Clasificar nuevo caso.	
	12	Recomendar algoritmos de descubrimiento.	

2.14. Tarjetas CRC.

Una tarjeta CRC (clase, responsabilidad y colaboración) representa una entidad del sistema, a las cuales asigna responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre clientes y equipo de desarrollo, en sesiones en las que se aplican

técnicas de grupos como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir de especificación de requisitos o historias de usuarios. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones (Escribano 2006).

Tabla 11: Tarjeta CRC ClasificationAlgorithm.

Clase: ClasificationAlgorithm	
Súper clase:	
Sub Clase(s):	
Responsabilidad: Es la encargada de poseer las características de un clasificador en general, su evaluación y la recomendación dada por él.	Colaboración: Clase: AbstractClassifier Clase: Evaluation

El resto de las tarjetas CRC se encuentran en el Anexo 2.

2.15. Diagrama de clases y paquetes.

A pesar de que la metodología XP no define la realización de diagramas de clases o diagramas de paquetes, se elaboraron para una mejor comprensión de la solución propuesta en esta investigación. Por las dimensiones del diagrama de clases fue necesario seccionarlo en 4 diagramas para su mejor visualización. Se muestra en la Figura 8 la descomposición por paquetes del sistema y posteriormente los diagramas de clases correspondiente a cada uno de estos (Figura 9, Figura 10 y Figura 11).

El diagrama de paquetes representa la estructura lógica de un sistema así como las dependencias o relaciones entre estas. El sistema debe definirse de forma simple y sin entrar en grandes detalles para que el diseñador realice un primer entregable del sistema. El diagrama de paquetes proporciona un modelo de los diferentes elementos que componen al sistema así como sus relaciones, además de permitir un diseño inicial sencillo del sistema.

Se presenta en la Figura 12 el diagrama de paquete de la API de Weka, aunque la metodología no define este tipo de diagrama, se expone el mismo debido a que se integra con el complemento y provee la utilización de los clasificadores sin necesidad de su implementación. Por lo que el estudio de su estructura es importante para esta investigación.

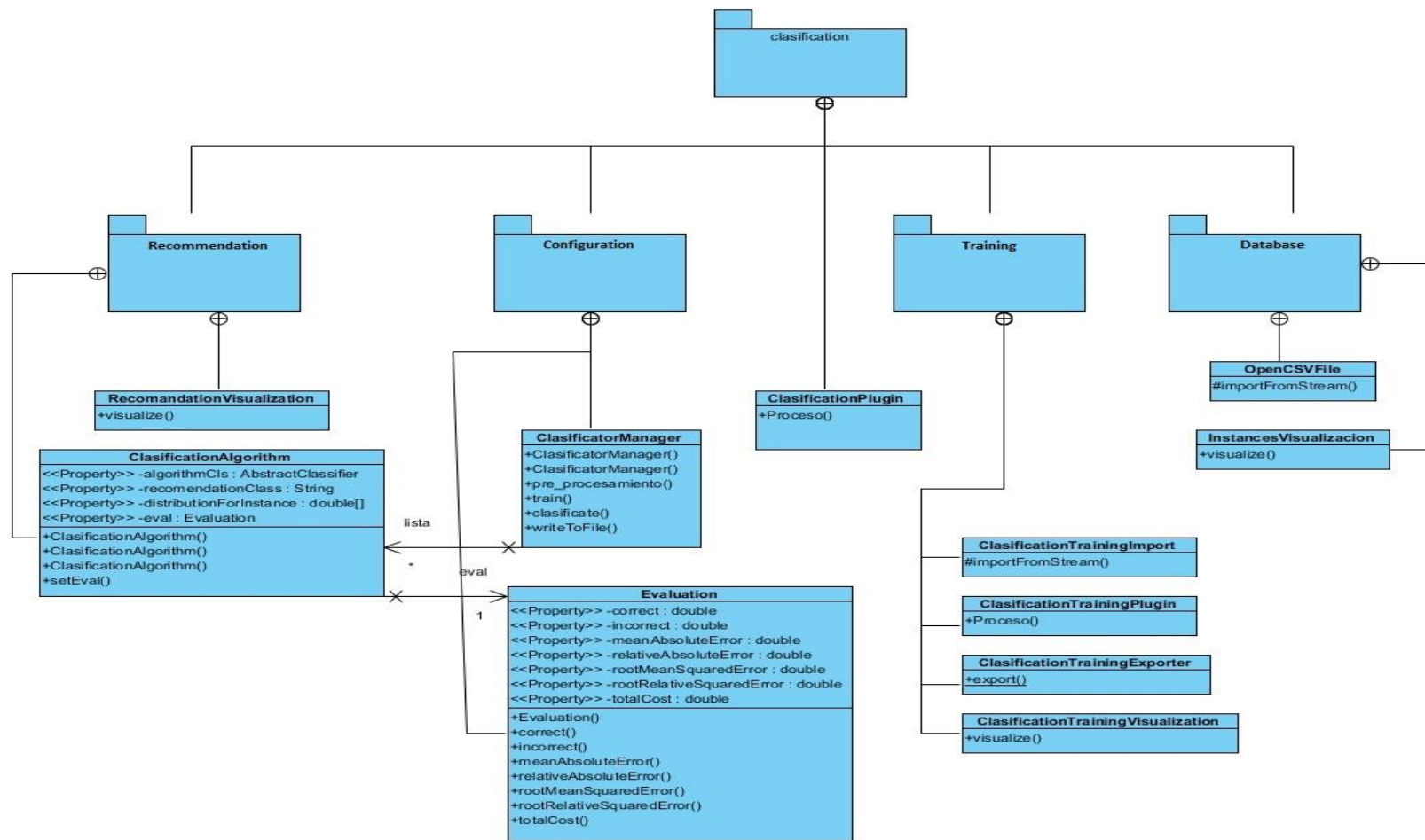


Figura 5: Diagrama de paquetes del sistema.

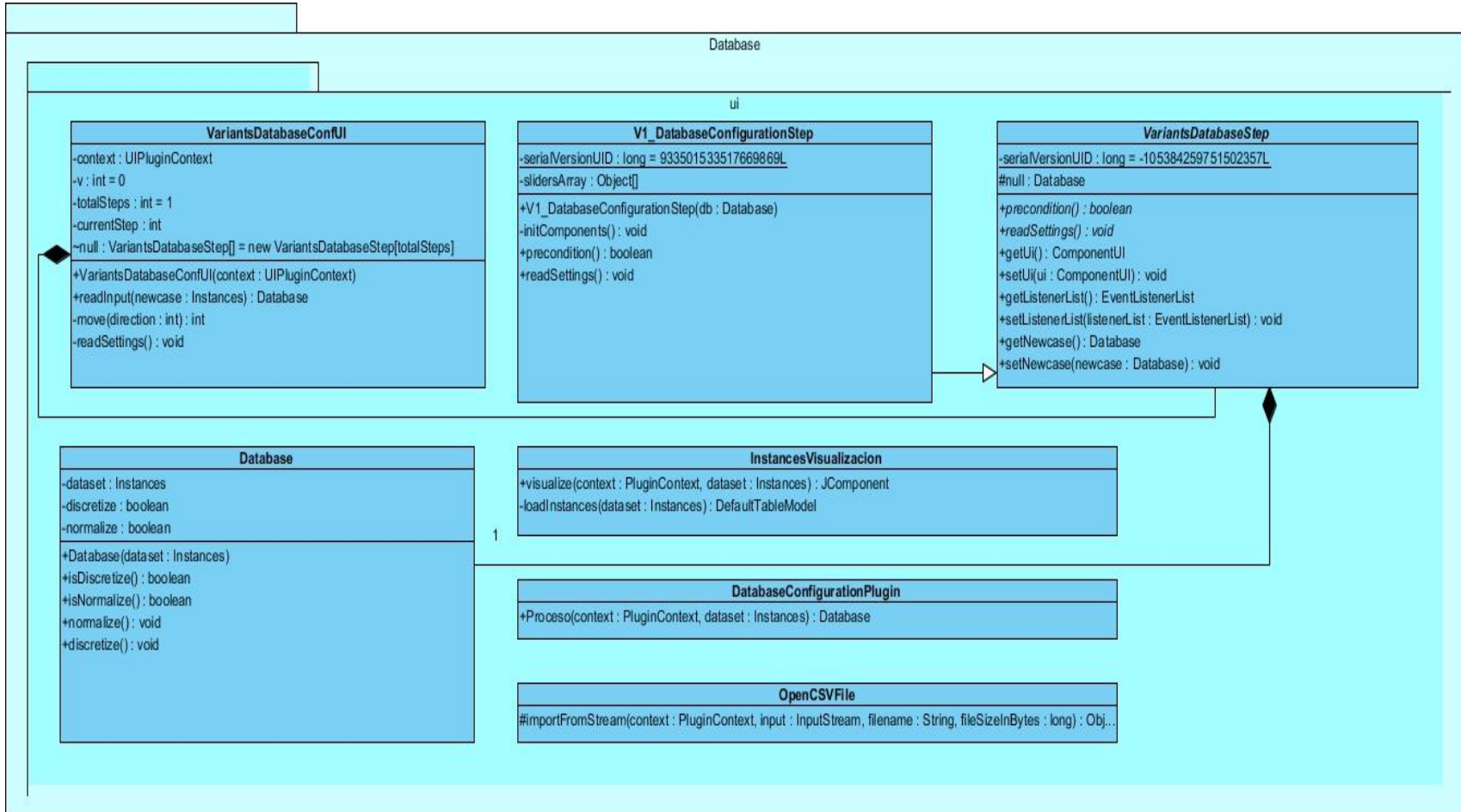


Figura 6: Diagrama de clases del paquete Database.

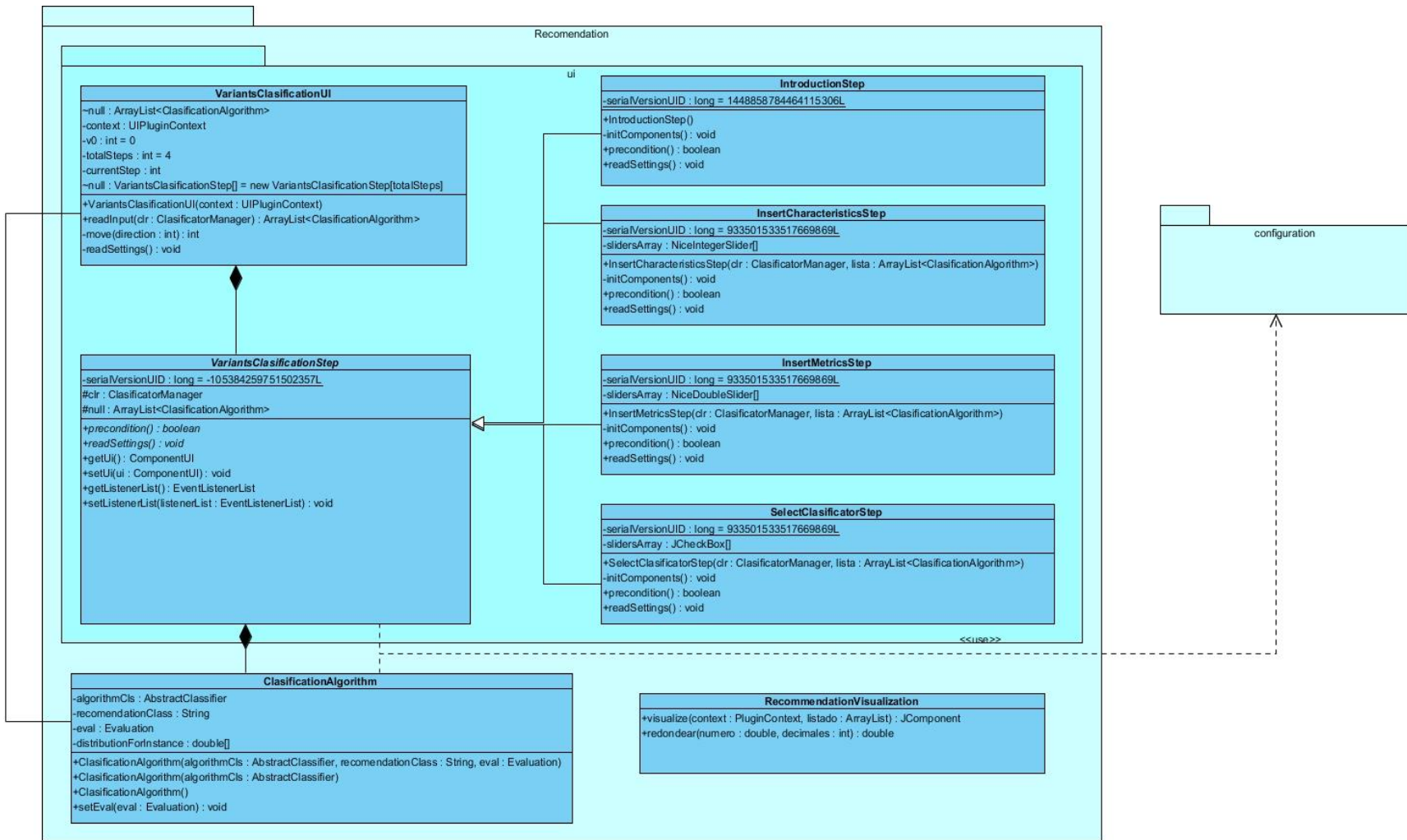


Figura 7: Diagrama de clases del paquete Recommendation.

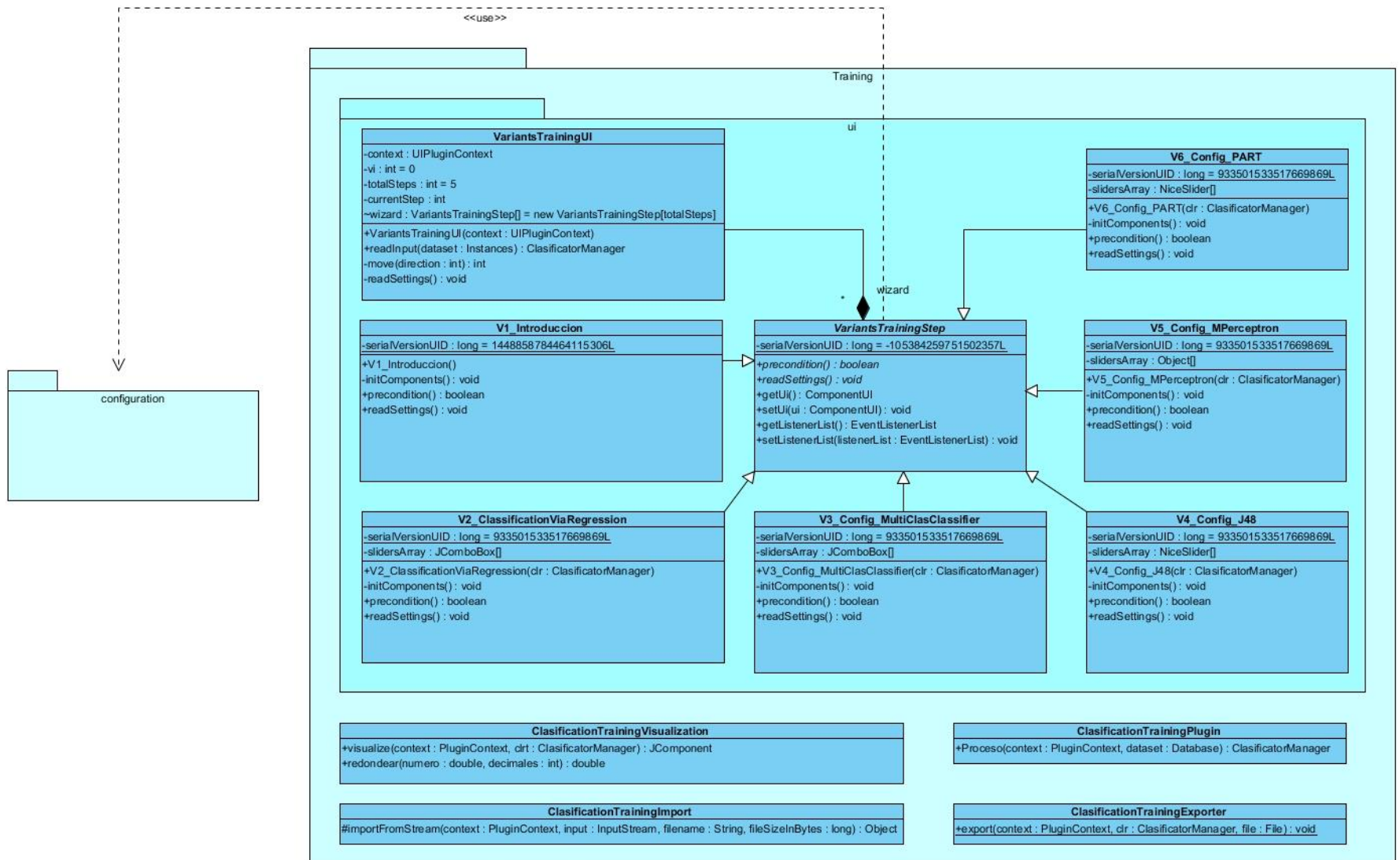


Figura 8: Diagrama de clases del paquete Training.

2.15.1. Diagrama de paquetes de la API de Weka.

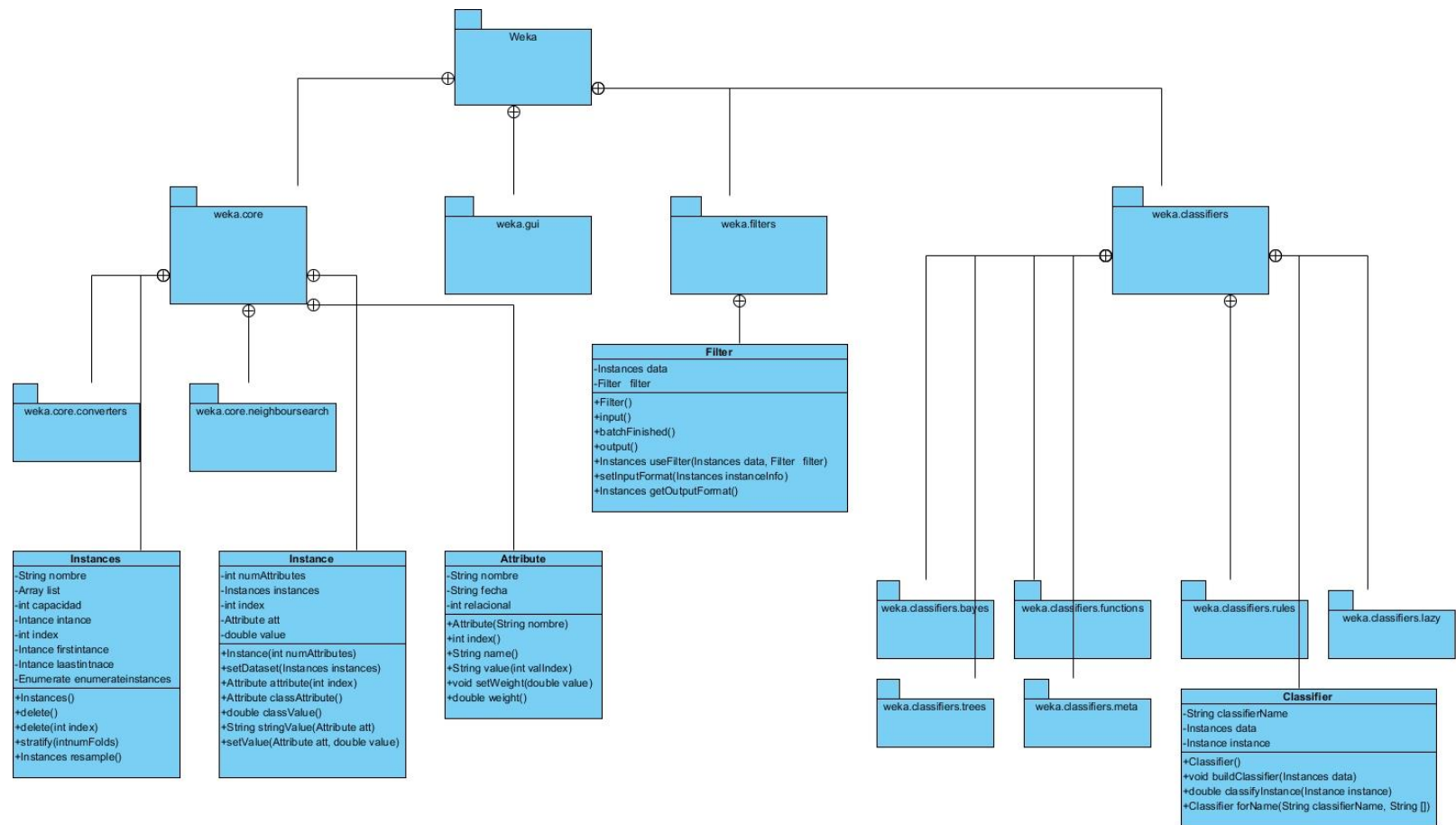


Figura 9: Diagrama de paquetes.

2.16. Arquitectura del sistema.

Una arquitectura software viene determinada por las diferentes instancias de cada tipo de componentes y conectores que la componen, y por una serie de enlaces (bindings) específicos que definen la unión de todas ellas formando una estructura. A esta estructura se le da el nombre de configuración y suele considerarse insertada en una jerarquía, pues toda entidad software, independientemente de su granularidad, dispone de una estructura que puede ser descrita mediante una arquitectura software (Montilva, Arapé, Colmenares 2003).

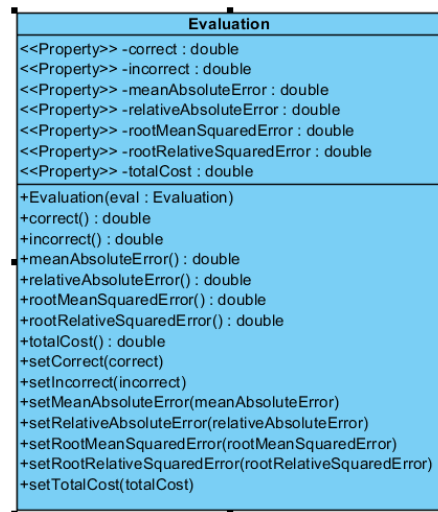
Al proponer la creación de un complemento para ProM como solución a la problemática planteada, se establece para este el uso de la arquitectura utilizada en el framework. Por esta razón se estableció entonces una arquitectura basada en componentes en correspondencia con la utilizada en ProM (*ProM 6* 2013).

2.17. Patrones de diseño.

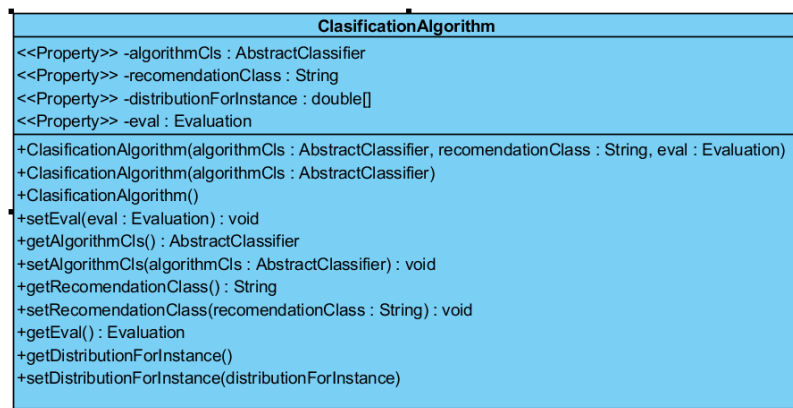
Para el diseño de la propuesta de solución que se tuvieron en cuenta los patrones GRASP y GoF. Los patrones GRASP (object-oriented design General Responsibility Assignment Software Patterns) describen cada uno de los principios principales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Mientras que los patrones GoF (Gang of Four) se descubren como una forma indispensable de enfrentarse a la programación.

Patrones GRASP.

Experto: La asignación de una responsabilidad a la clase que cuenta con la información necesaria para cumplir una funcionalidad en específico, permite tener un código correcto y funcional a la hora de utilizarlo. En el complemento se cuenta con clases que cumplen con este patrón de diseño como por ejemplo la clase Evaluation, que encierra todas las funcionalidades necesarias para evaluar a los clasificadores.



Creador: Patrón para las clases que tienen como responsabilidad crear instancias de otras para poder realizar completamente el funcionamiento del proceso. Con este principio se creó la clase `ClassificationAlgorithm` que se encarga de crear instancias de las clases `Evaluation` y `AbstractClassifier` para su funcionamiento.



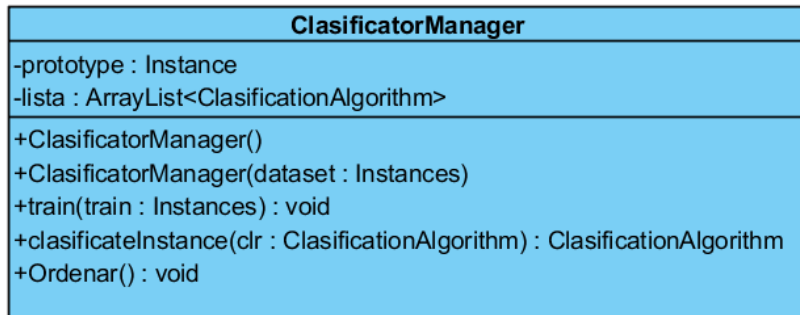
Bajo Acoplamiento: Con este patrón se persigue es lograr poca dependencia entre las clases, evitando que cualquier cambio que pueda suceder durante el desarrollo de la aplicación tenga gran repercusión en el sistema. Por ejemplo la clase `Database` solo depende de la clase `Instances` para realizar sus funciones.

Database
-dataset : Instances -discretize : boolean -normalize : boolean
+Database(dataset : Instances) +isDiscretize() : boolean +isNormalize() : boolean +normalize() : void +discretize() : void +getDataset() : Instances +setDataset(dataset : Instances) : void +setDiscretize(discretize : boolean) : void +setNormalize(normalize : boolean) : void

Alta cohesión: Define que generalmente una clase que este altamente cohesionada tiene bajo acoplamiento. Esto se debe a que clase ya que posee la menor cantidad de dependencias posibles con otras clases y a su vez posee responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. La clase VariantsTrainingUI posee la responsabilidad de construir las ventanas de configuración de los distintos complementos del sistema, relacionándose con un reducido número de clases sin sobrecargarse con métodos que no tiene que ver con esta funcionalidad.

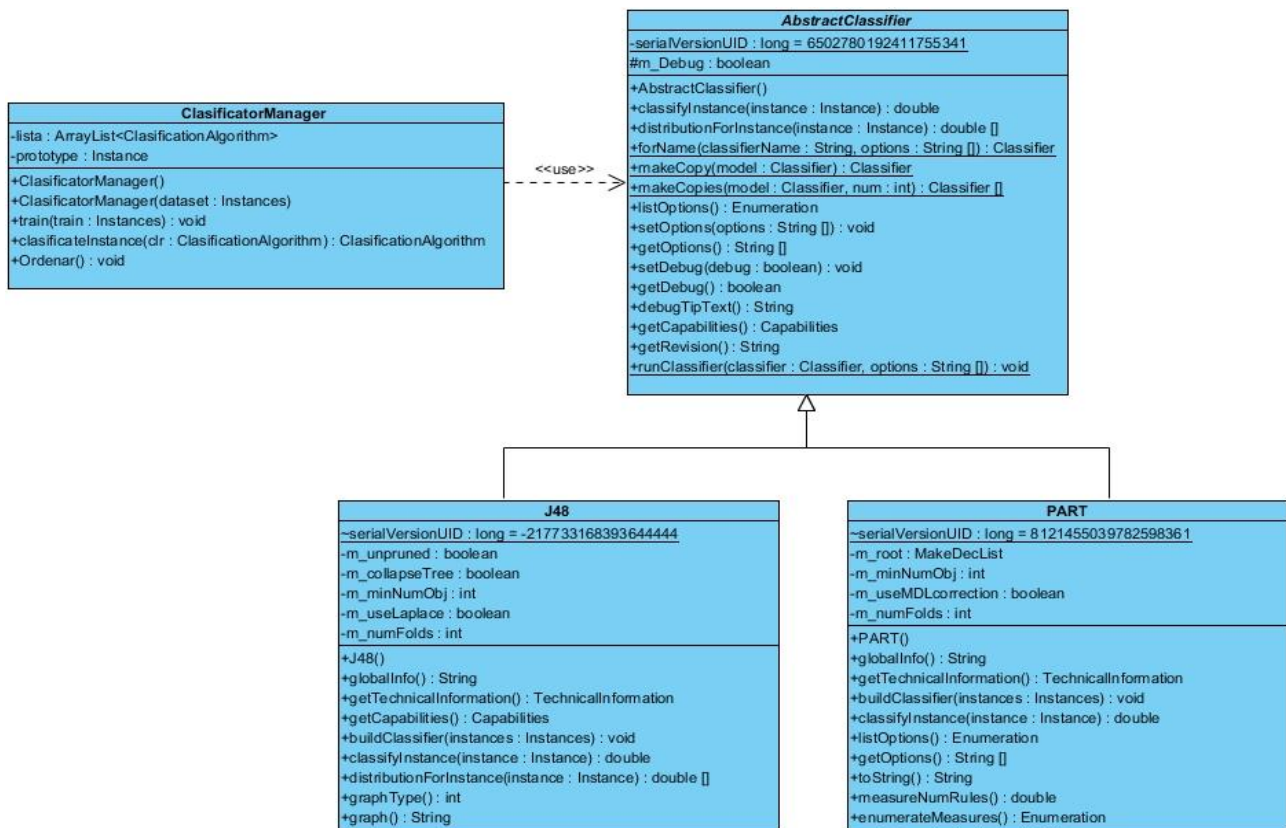
VariantsTrainingUI
-context : UIPluginContext -vi : int = 0 -totalSteps : int = 6 -currentStep : int ~wizard : VariantsTrainingStep[] = new VariantsTrainingStep[totalSteps]
+VariantsTrainingUI(context : UIPluginContext) +readInput(dataset : Instances) : ClasificadorManager -move(direction : int) : int -readSettings() : void

Controlador: Este patrón sugiere el uso de clases controladoras para separar la lógica de negocio de la capa de presentación. En el framework ProM la clase Plugin es la encargada de controlar y definir los complementos integrados en el sistema. Sin embargo por lo general se cuenta con clases internas en los distintos complementos que cumplen esta responsabilidad. Para el sistema propuesto la clase que cumplen con este patrón es: “ClasificadorManager”.



Patrones GoF.

Estrategia: Encapsula algoritmos relacionados en clases que son subclases de una superclase común. Esto permite la selección de un algoritmo que varía según el objeto y también le permite la variación en el tiempo. Siguiendo este patrón se evidencia en las clases MultiClassClassifier, J48 y el resto de los clasificadores implementados que heredan de AbstractClassifier. Esta a su vez interactúa con ClasificadorManager que es la encargada de entrenar los clasificadores independientemente del tipo que seas.



Decorador: Este patrón se encarga de añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. La aplicación de este patrón quedó evidenciado con el uso de las anotaciones implementadas en ProM, que por su estructura brinda un conjunto de anotaciones para definir tipos de complementos (para importar, exportar, procesar y visualizar) sin hacer uso de la herencia. Un ejemplo de esto lo son las clases: `ClassificationPlugin`, `ClassificationTrainingVisualization`, `ClassificationTrainingImport` y `ClassificationTrainingExporter`, entre otras.

Fachada: Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. Se utiliza al unificar las interfaces en una sola ventana, manejando esta varias interfaces. La clase `ClassifierManager` cumple con este patrón debido a que es la que establece una puerta de enlace entre la API de Weka y el resto de las clases del sistema. Solo se accede a las funcionalidades de la API de Weka mediante esta clase.

2.18. Conclusiones Parciales.

En el presente capítulo se puede concluir con las siguientes afirmaciones:

- Los clasificadores que obtuvieron mayor efectividad frente a una base de conocimiento con información sobre el rendimiento de los algoritmos de descubrimiento fueron: Perceptrón Multicapa, Classification Via Regression, MultiClass Classifier, PART y J48.
- La correcta planificación del sistema permitió la obtención de 12 requerimientos funcionales y 5 no funcionales así como su descripción según la metodología XP.
- El diseño del sistema quedó representado por los artefactos que brinda la metodología XP en esta fase, apoyándose además en diagramas de clases y de paquetes del sistema para un mejor entendimiento.

3. IMPLEMENTACIÓN Y PRUEBA

Para el desarrollo del complemento es de vital importancia la implementación y validación del sistema propuesto, para dar cumplimiento a los objetivos planteados. Para alcanzar dicho propósito en este capítulo se describen los estándares de codificación, las tareas de ingeniería, las pruebas realizadas a la aplicación con el objetivo de verificar si se cumplieron las tareas de ingeniería planificadas para el desarrollo de la aplicación y la validación de la propuesta de solución.

3.1. Implementación

Para la implementación del sistema se utilizó el IDE NetBeans y el lenguaje de programación Java. En la fase de planificación se definieron las funcionalidades necesarias para el desarrollo del complemento. En esta fase de implementación esas funcionalidades definidas en las Historias de Usuarios se dividen en tareas más pequeñas, denominadas tareas de ingeniería, con el objetivo de realizar un análisis con mayor detalle y realizar una estimación real de su tiempo de desarrollo.

3.2. Estándar de codificación.

Debido a que la metodología XP considera al código como propiedad colectiva, es necesario tener en cuenta un estándar de codificación. Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se recomienda establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada (Escribano 2006).

- **Nomenclatura de las clases**

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de poseer nombre compuesto se comienzan todo con letra mayúscula.

- **Nomenclatura de las funciones**

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula. Si son compuestos por más de un nombre se continúa con letra inicial mayúscula.

- **Nomenclatura de las variables**

El nombre a emplear para las variables se escribe con la primera palabra en minúscula.

- **Normas de comentario**

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

3.2.1. Ejemplos de código implementado.

A continuación se presentan fragmentos de código correspondientes al sistema implementado.

Operadores:

- Los operadores binarios que se encuentran entre dos valores como +, -, =, <=, ==, => deben tener un espacio antes y después del operador, para facilitar la lectura.

```
BayesNet aux = new BayesNet();
```

- Los operadores unarios, que operan en un solo valor, tales como ++, no debería tener un espacio entre el operador y el número variable.

```
for (int i = 0; i < list.size(); i++)
```

Estructuras de control:

- Las sentencias de control debe tener un espacio entre la palabra de control y paréntesis de apertura para distinguirlas de las llamadas a funciones.
- Siempre se utilizan las llaves, incluso en situaciones en las que son opcionales ya que aumentan la legibilidad y disminuye la probabilidad de errores lógicos.

```
if (slidersArray[2].getSelectedIndex() == 0) {
    aux.setEstimator(new SimpleEstimator());
} else if (slidersArray[2].getSelectedIndex() == 1) {
    aux.setEstimator(new BayesNetEstimator());
} else if (slidersArray[2].getSelectedIndex() == 2) {
    aux.setEstimator(new BMAEstimator());
} else if (slidersArray[2].getSelectedIndex() == 3) {
    aux.setEstimator(new MultiNomialBMAEstimator());
}
```

Ciclos for:

- Las sentencias for debe tener un espacio entre la palabra de control y paréntesis de apertura para distinguirlas de las llamadas a funciones y otro espacio para separar el paréntesis de cierre y la llave de apertura del ciclo.
- La llave de cierre de la sentencia debe ir sola y una línea debajo del segmento de código interno del ciclo.

```
for (ClasificationAlgorithm lista1 : lista) {  
    lista1.getAlgorithmCls().buildClassifier(train);  
    ev = new weka.classifiers.Evaluation(train);  
    ev.crossValidateModel(lista1.getAlgorithmCls(), train, 10, new Random(1));  
    lista1.setEval(ev);  
}
```

3.3. Desarrollo de las iteraciones.

En el plan de iteraciones se definieron 3 en total donde cada una de estas contaba con el desarrollo de 4 HU para un total de 12 a desarrollar. Las HU se descomponen en tareas de programación o tareas de ingeniería asignada a un miembro del equipo responsable de su cumplimiento.

Iteración 1

Esta iteración tiene como objetivo dar cumplimiento a las HU 1, 2, 3, 4 las cuales comprenden importar base de casos, pre-procesar base de casos, configurar clasificadores, entrenar clasificadores.

Tabla 12: Tarea No.1 de la HU No 1

Tarea de Ingeniería	
Número Tarea: 1	Número Historia de Usuario: 1
Nombre Tarea: Investigar cómo se importa una fichero en la plataforma ProM.	
Tipo de Tarea: Investigación.	Puntos Estimados: 1
Fecha Inicio: 01/03/2014	Fecha Fin: 2/03/2014
Programador Responsable: Michel Hernández Pajaro.	

Descripción: Se realiza un estudio de cómo se importa un fichero en la plataforma ProM tanto desde una dirección física en la estación de trabajo en formato CSV o desde un gestor de base de datos.

Iteración 2

Esta iteración tiene como objetivo dar cumplimiento a las HU 5, 6, 7, 8 las cuales comprenden evaluar clasificadores, exportar clasificadores entrenados, importar clasificadores entrenados, visualizar clasificadores entrenados.

Tabla 13: Tarea No.2 de la HU No.6

Tarea de Ingeniería	
Número Tarea: 12	Número Historia de Usuario: 6
Nombre Tarea: Implementar la funcionalidad "Exportar clasificadores entrenados".	
Tipo de Tarea: Desarrollo.	Puntos Estimados: 1
Fecha Inicio: 04/03/2014	Fecha Fin: 05/04/2014
Programador Responsable: Raciel Roche Escobar.	
Descripción: Se realizó la implementación de la funcionalidad "Exportar clasificadores entrenados".	

Iteración 3

Esta iteración tiene como objetivo dar cumplimiento a las HU 9, 10, 11, 12 las cuales comprenden crear nuevo caso, seleccionar clasificadores, clasificar nuevo caso, recomendar algoritmos de descubrimiento.

Tabla 14: Tarea No.1 de la HU No.9

Tarea de Ingeniería	
Número Tarea: 18	Número Historia de Usuario: 9
Nombre Tarea: Implementar la funcionalidad "Crear nuevo caso".	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1
Fecha Inicio: 15/03/2014	Fecha Fin: 19/04/2014
Programador Responsable: Raciel Roche Escobar	

Descripción: Se realiza la implementación de la funcionalidad “Crear nuevo caso”.

El resto de las tareas de ingeniería se encuentran en el Anexo 3.

3.4. Pruebas a desarrollar.

Una parte importante de la metodología XP son las pruebas, mediante las cuales los desarrolladores prueban constantemente el sistema para detectar las posibles fallas que puedan ocurrir. Esta práctica contribuye no solo a elevar la calidad del sistema que se quiere construir, además verifica que el proceso de implementación se realiza sobre funcionalidades con un correcto funcionamiento.

3.4.1. Pruebas Unitarias.

Para las pruebas de Unitarias se utilizó el framework JUnit, el cual brinda un conjunto de librerías que permitirán la integración con el IDE de desarrollo. JUnit permite la realización de las pruebas a los métodos de las clases implementadas en ambas capas. Las pruebas fueron aplicadas a las funcionalidades más importantes.

Funcionalidades

- Importar Base de Casos.
- Entrenar Clasificadores.
- Crear nuevo caso.
- Recomendar algoritmo de descubrimiento.

La siguiente imagen muestra las pruebas realizadas a las clases OpenCSVFile y DatabaseConfigurationPlugin las cuales contienen los métodos necesarios para importar y configurar una base de casos en el sistema. Los resultados fueron satisfactorios.

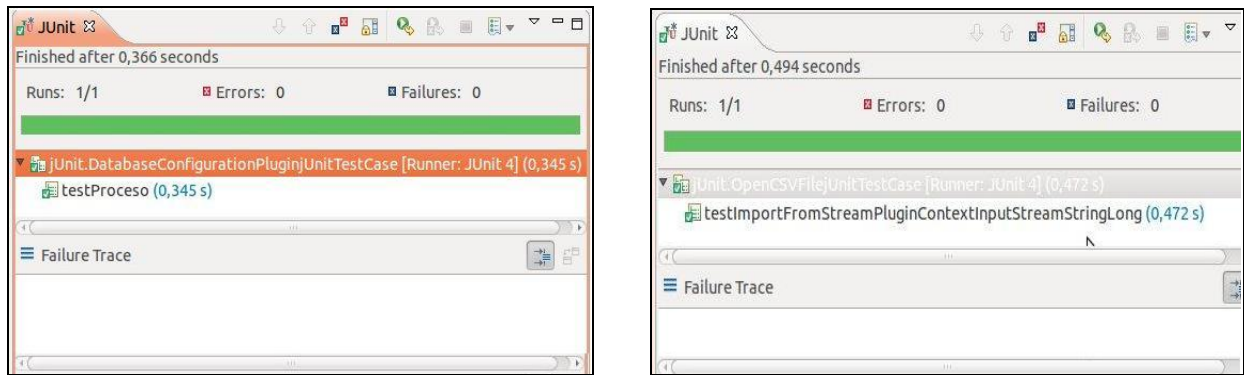


Figura 10: Resultados de las pruebas a las clases OpenCSVFile y DatabaseConfigurationPlugin.

El resto de las pruebas de las pruebas Unitarias se encuentran en el Anexo 6.

Una vez definidas las pruebas y utilizando la estrategia definida para usar el framework JUnit, se probaron las funcionalidades principales de cada una de las clases implementadas. En la primera ejecución de las pruebas se detectaron 10 errores y 4 fallos contenidos en 8 pruebas fallidas de 10. Se corrigieron estos errores y fallos además de los encontrados en posteriores ejecuciones de las pruebas hasta la supresión de los mismos.

3.4.2. Pruebas de Aceptación.

Las pruebas de aceptación son creadas basándose en las historias de usuarios. Dichas pruebas son consideradas como pruebas de caja negra y los clientes son los responsables de verificar que el resultado de estas pruebas sean los correctos (Letelier, Penadés 2006).

A continuación se presentan un conjunto de casos de prueba destinados a comprobar el funcionamiento del sistema.

Tabla 15: Caso de Prueba de Aceptación Clasificar nuevo caso.

Caso de Prueba de Aceptación	
Código Caso de Prueba: CRE1111	Nombre Historia de Usuario: Clasificar nuevo caso.
Nombre de la persona que realiza la prueba: Raciél Roche Escobar.	

Descripción de la Prueba: El sistema debe permitirle al usuario clasificar nuevo caso.
Condiciones de Ejecución: Se ejecuta la aplicación satisfactoriamente. Se importa los clasificadores entrenados o se importa una base de casos y se entrenan los clasificadores. Es construido el nuevo caso. Son seleccionados los clasificadores.
Entrada / Pasos de ejecución: El usuario debe seleccionar la opción clasificar.
Resultado Esperado: El nuevo caso es clasificado.
Evaluación de la Prueba: Satisfactoria

El resto de los casos de prueba se encuentran en el Anexo 4.

3.4.3. Resultados obtenidos.

En la metodología XP, la aplicación de pruebas se realiza de forma iterativa para comprobar la correcta implementación de las historias de usuario (Letelier, Penadés 2006). Fueron aplicadas un total de 12 casos de prueba de aceptación que se distribuyeron en 3 iteraciones. Fueron probadas 12 funcionalidades donde se detectaron un total 6 no conformidades. Se encontraron en la primera iteración 4 errores ortográficos, 1 error por demora de tiempo de respuesta en el entrenamiento de los clasificadores y 1 error funcional al importar una base de casos. Estos problemas fueron corregidos en la segunda iteración y no fueron detectadas nuevas no conformidades en la iteración restante.

A continuación se presenta una gráfica con el resumen de las no conformidades detectadas.

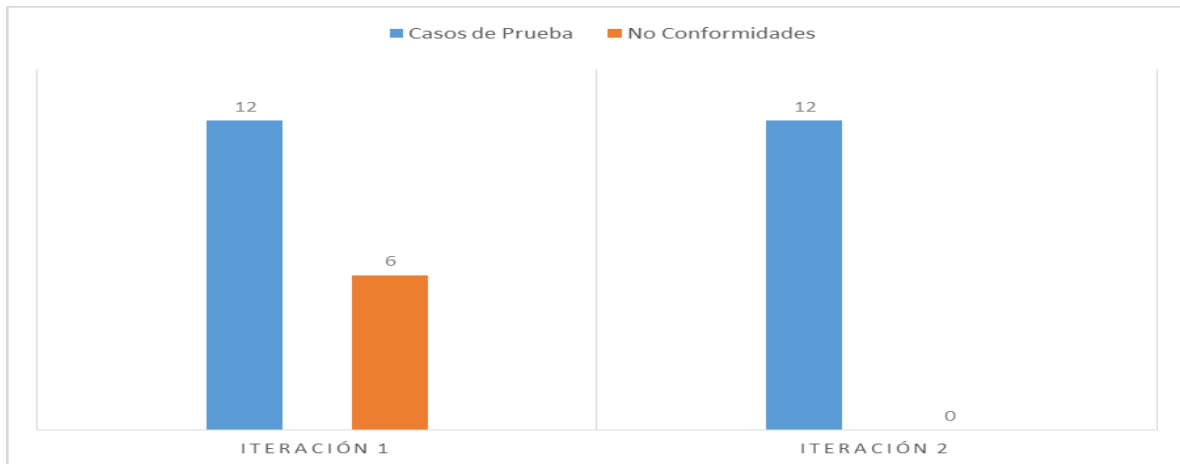


Figura 11: Resumen de las no conformidades detectadas.

3.5. Validación de la propuesta de solución.

La validación de la propuesta implementada se divide en dos momentos importantes. En un primer momento se contrastan los resultados obtenidos por los clasificadores implementados en el complemento para ProM y los clasificadores de WEKA. Esta validación permite garantizar que el rendimiento de los algoritmos de clasificación del complemento, este en correspondencia con el rendimiento de los algoritmos de Weka. Seguidamente, en un segundo momento, se realiza una comparación entre los tiempos de la selección por Evaluación Empírica y por los clasificadores implementados. Con esta validación se intenta demostrar que existe una disminución real del tiempo requerido para la selección de algoritmos de descubrimiento.

3.5.1. Validación respecto a Weka.

Debido a que los clasificadores implementados fueron obtenidos por la integración del complemento con la API de Weka, el rendimiento de estos debe comportarse de manera semejante para ambas herramientas. Para comprobar la similitud de estos resultados, se seleccionó una base de casos (BC2) que poseía un total de 795 tuplas referentes a la aplicación de 4 algoritmos de descubrimiento sobre 74 registros de eventos y su evaluación mediante 4 métricas de calidad (véase epígrafe 2.2).

Con esta base de casos se entrenaron los clasificadores en ambas herramientas y se evaluaron a través de una Validación cruzada (10 particiones). Se tuvo en cuenta principalmente el

número de instancias correctamente clasificadas (ICC) como parámetro definitorio, además del tiempo de construcción del modelo (TE). A continuación se muestran el resultado de dicha comparación.

Tabla 16: Resultado de la comparación

Algoritmo	Weka		Complemento	
	TE	ICC	TE	ICC
MultiClassClassifier	1,04	795	1.65	795
MultiLayer Perceptron	2,86	794	5.42	794
ClassificationViaRegression	0,1	792	0.21	792
PART	0,01	792	0.10	792
J48	0,01	792	0.03	792

Como se puede apreciar el número de instancias correctamente clasificadas coincide para cada clasificador en las dos herramientas. Sin embargo aunque el tiempo de construcción del modelo difiere para cada caso, no es relevante debido a que esta pequeña diferencia no afecta la calidad de los resultados. Esta diferencia puede ser causada por factores externos que estén afectado el rendimiento de la estación de trabajo en el momento de las pruebas.

3.5.2. Validación respecto a la Evaluación Empírica.

3.5.2.1. Disminución del tiempo.

Como se explica en el capítulo 1, la Evaluación Empírica consiste en probar un conjunto de algoritmos de descubrimiento sobre un mismo registro de eventos. Posteriormente se evalúan los modelos obtenidos con una serie de métricas de calidad y se seleccionan los algoritmos que hayan brindado los mejores resultados. La ejecución de cada métrica y cada algoritmo tiene un costo computacional asociado que varía en función de sus características. El tiempo requerido para la Evaluación Empírica de algoritmos, se determina por la suma del tiempo de ejecución de cada uno de los algoritmos de descubrimiento evaluados y el tiempo de ejecución de las métricas a utilizar.

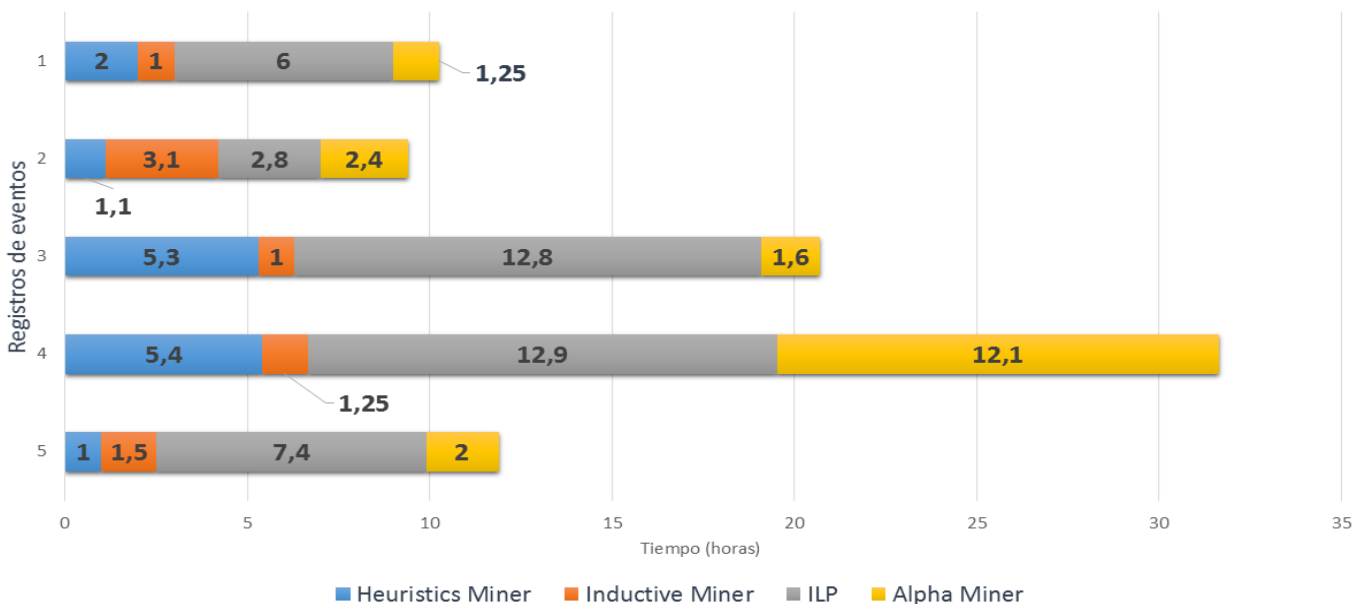
Partiendo de que la Evaluación Empírica es la técnica de Minería de Procesos más utilizada para la selección de algoritmos de descubrimiento de procesos (Claes, Poels 2013), se determinó que la validación del sistema debía estar enfocada a disminuir el tiempo requerido

por esta técnica. Por esta razón se decidió comparar los tiempos requeridos para la selección por Evaluación Empírica y los tiempos de la selección utilizando los clasificadores del complemento.

Para realizar este procedimiento se seleccionaron cinco registros de eventos con la misma estructura de BC2, o sea 8 características del registro de eventos, 4 algoritmos de descubrimiento y su evaluación mediante 4 métricas de calidad. El Anexo 5 muestra en detalle los registros de eventos seleccionados para estas pruebas.

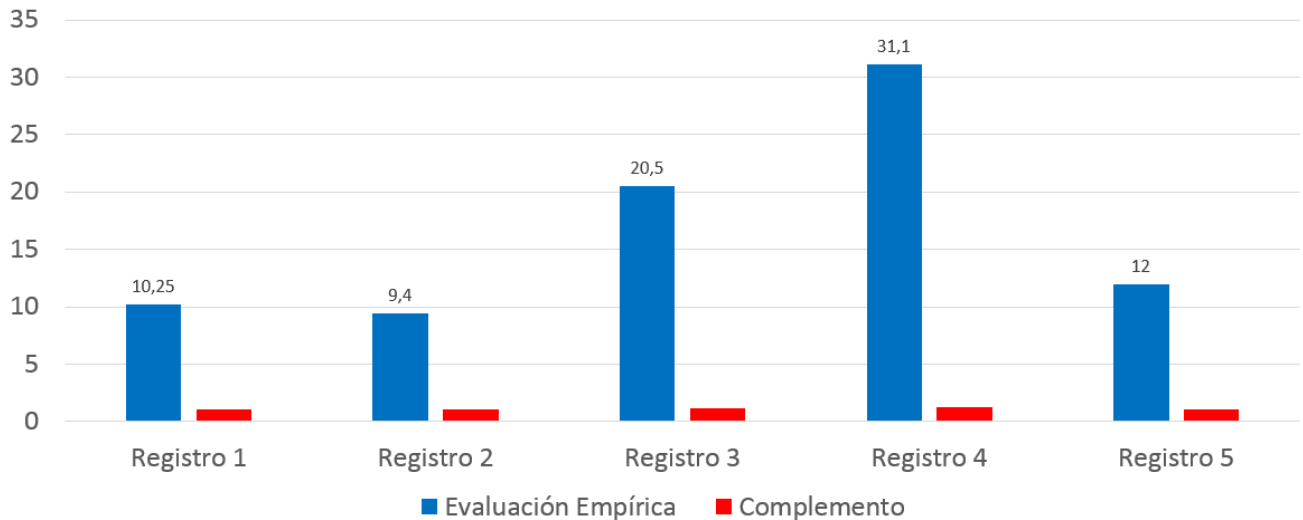
Se procedió entonces a la medición del tiempo consumido por la Evaluación Empírica al ser aplicado a los registros de eventos antes mencionados. Con los resultados obtenidos se conformó la siguiente tabla.

Tabla 17: Tiempo de la Evaluación Empírica.



Simultáneamente a la Evaluación Empírica se determinaron los tiempos necesarios para la recomendación por el complemento implementado para ProM. Luego de esto se contrastaron los tiempos obtenidos para verificar si la disminución de tiempo requerido era efectiva. La tabla que se muestra a continuación contiene los tiempos necesarios para la recomendación en cada registro de eventos en ambas alternativas.

Tabla 18: Tiempos de Evaluación Empírica y los clasificadores del sistema.



Registro de Eventos	Evaluación Empírica	Complemento
1	10h 14m 15s	10s
2	9h 13m 24s	9s
3	20h 28m 30s	15s
4	31h 1m 35s	40s
5	11h 55m 51s	10s

Se puede apreciar que existe una disminución notable en el tiempo requerido para la selección de algoritmos de descubrimiento con el uso del sistema implementado. Además es válido mencionar que el proceso de clasificación consta de dos fases, el entrenamiento de los clasificadores y la clasificación del nuevo caso. El tiempo requerido para el entrenamiento o construcción del modelo es frecuentemente más elevado que el estimado para la clasificación. Esto queda contrarrestado debido a que el entrenamiento solo se realiza cuando el nuevo caso a clasificar no posee la misma estructura que la base de casos utilizada para el entrenamiento, por lo que su reutilización es un factor a tener en cuenta. Para la base de casos empleada el tiempo para la construcción del modelo fue de 1 minuto.

Como se puede apreciar la disminución del tiempo requerido para la recomendación de algoritmos de descubrimiento es considerable, quedando validado el sistema en este aspecto.

3.5.2.2. Efectividad de la recomendación.

La Evaluación Empírica, aunque es muy costosa en cuanto a tiempo y recursos computacionales, permite recomendar el mejor algoritmo de descubrimiento para el registro de eventos en cuestión. Este aspecto es de vital importancia para poder validar la propuesta implementada, ya que reducir el tiempo para la selección del algoritmo no es suficiente, el algoritmo seleccionado debe ser semejante al recomendado por la Evaluación Empírica. En función de lo antes expuesto, se evaluaron los resultados obtenidos por ambas técnicas para comprobar la efectividad del complemento.

El sistema realizado recomienda un algoritmo de descubrimiento por cada algoritmo de clasificación implementado. Actualmente el complemento cuenta con 8 algoritmos de clasificación que pueden recomendar diferentes algoritmos de descubrimiento. La efectividad está dada por la coincidencia que exista entre el algoritmo de descubrimiento recomendado por la Evaluación Empírica y el que haya sido recomendado por la mayor cantidad de algoritmos de clasificación del complemento.

La estructura de un nuevo caso en el complemento está dada por las características del registro de eventos y el valor esperado de las métricas de calidad. Este último factor puede variar e incidir en el resultado de un clasificador. En relación a esto se determinaron varios juegos de datos para las métricas de calidad que permite mostrar cierto grado de variabilidad en los resultados de los clasificadores del complemento. Se construyeron por cada registro de eventos tres nuevos casos donde para el valor esperado de las métricas de calidad se introdujeron: el valor máximo permisible, la moda y el promedio.

A continuación se muestran las tablas con la recomendación otorgada por la Evaluación Empírica y la brindada por los clasificadores del complemento. Para los registros de eventos en el que coincidieron las recomendaciones dadas a los tres nuevos casos construidos por este, solo se muestra un solo resultado.

Registro de Eventos # 1 Recomendado por Evaluación Empírica: Alpha.

Algoritmo de clasificación	Recomendado por el sistema.
ClassificationViaRegression	Alpha
FilteredClassifier	Alpha
J48	Alpha
Logistic	Heuristics

MultiLayer Perceptron	Alpha
MultiClassClassifier	Alpha
PART	Alpha
Simple Logic	Alpha

Registro de Eventos # 2 Recomendado por Evaluación Empírica: Alpha.

Algoritmo de clasificación	Recomendado por el sistema.
ClassificationViaRegression	Alpha
FilteredClassifier	Alpha
J48	Alpha
Logistic	Heuristics
MultiLayer Perceptron	Alpha
MultiClassClassifier	Alpha
PART	Alpha
Simple Logic	Alpha

Registro de Eventos # 3 Recomendado por Evaluación Empírica: Alpha.

Algoritmo de clasificación	Recomendado por el sistema.
ClassificationViaRegression	Alpha
FilteredClassifier	Alpha
J48	Alpha
Logistic	Heuristics
MultiLayer Perceptron	Alpha
MultiClassClassifier	Alpha
PART	Alpha
Simple Logic	Alpha

Registro de Eventos # 4 Recomendado por Evaluación Empírica: Heuristics.

Algoritmo de clasificación	Recomendado por el sistema.		
	Moda	Máximo valor permisible	Promedio
ClassificationViaRegression	Alpha	Inductive	Heuristics
FilteredClassifier	Alpha	Alpha	Alpha
J48	Alpha	Inductive	Alpha
Logistic	Heuristics	Heuristics	Heuristics
MultiLayer Perceptron	Alpha	Alpha	Alpha

MultiClassClassifier	Alpha	Alpha	Heuristics
PART	Alpha	Inductive	Alpha
Simple Logic	Alpha	Alpha	Inductive

Registro de Eventos # 5 Recomendado por Evaluación Empírica: Alpha.

Algoritmo de clasificación	Recomendado por el sistema.
ClassificationViaRegression	Alpha
FilteredClassifier	Alpha
J48	Alpha
Logistic	Heuristics
MultiLayer Perceptron	Alpha
MultiClassClassifier	Alpha
PART	Alpha
Simple Logic	Alpha

En correspondencia con el criterio de similitud seleccionado, para los registros 1, 2, 3 y 5, los algoritmos recomendados por Evaluación Empírica y los recomendados por la mayoría de los algoritmos de clasificación coinciden. Para el registro #4 los resultados de evaluar los distintos juegos de datos para las métricas de calidad, mostraron diferentes resultados para los clasificadores. Para el resto de los registros de eventos coincidieron los algoritmos recomendados para los distintos valores introducidos en las métricas de calidad.

De las pruebas realizadas se puede concluir que para 4 de los 5 registros de eventos analizados, se recomendó el algoritmo idóneo como primera opción. Lo que brinda un porcentaje elevado de efectividad al sistema. Las métricas de calidad pueden incidir en el resultado de la clasificación, por lo que cuando se desconozcan sus valores es recomendable utilizar el promedio como valor de sustitución. Por lo antes expuesto quedó validado el sistema implementado tanto en la disminución del tiempo requerido para la selección de algoritmos de descubrimiento, como a la efectividad de la selección.

3.6. Conclusiones Parciales.

En el presente capítulo se puede concluir con las siguientes afirmaciones:

- La implementación del complemento quedó plasmada en 3 iteraciones, conformadas por 12 historias de usuario y reflejadas en 24 tareas de ingeniería.

- El complemento desarrollado para ProM, basado en los clasificadores que demostraron mayor efectividad en el estudio realizado, permitió recomendar algoritmos de descubrimientos para registros de eventos.
- La solución desarrollada disminuye los tiempos requeridos para la recomendación de algoritmos de descubrimiento, con respecto a la Evaluación Empírica, sin afectar su efectividad.

4. CONCLUSIONES

Con la realización de este trabajo se ha cumplido con los objetivos propuestos. Se llegó además a las siguientes conclusiones:

- La investigación realizada permitió el análisis de las principales características que afectan el rendimiento de los algoritmos de descubrimiento.
- Los clasificadores que obtuvieron mayor efectividad frente a una base de conocimiento con información sobre el rendimiento de los algoritmos de descubrimiento fueron: Perceptrón Multicapa, Classification Via Regression, MultiClass Classifier, PART y J48.
- El complemento desarrollado para ProM, basado en los clasificadores que demostraron mayor efectividad en el estudio realizado, permitió recomendar algoritmos de descubrimientos para registros de eventos.
- La solución desarrollada disminuye los tiempos requeridos para la recomendación de algoritmos de descubrimiento, con respecto a la Evaluación Empírica, sin afectar su efectividad.

5. RECOMENDACIONES

Como resultado del proceso de investigación e implementación de la solución propuesta se recomiendan los siguientes elementos:

- Realizar un estudio de las características de los registros de eventos que afectan a los algoritmos de clasificación para seleccionar algoritmos clasificación en dependencia del registro de eventos.
- Aplicar la propuesta desarrollada a registros de eventos provenientes de entornos reales.
- Aplicar la propuesta desarrollada sobre bases de casos con diferentes características.

6. BIBLIOGRAFÍA

AALST, WMPVD, 2011a, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Heidelberg.

AALST, WMPVD, 2011b, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Heidelberg.

ARCIERI, F., STILO, G. and TALAMO, M., 2011, Process Mining Manifesto. *Lecture Notes in Business Information Processing*. 2011. Vol. 99, p. 169–194.

BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar, MARTÍNEZ, José Sáez and MOLINA, Jesús J. García, 1999, *El lenguaje unificado de modelado* [online]. Addison-Wesley. [Accessed 3 June 2014]. Available from: <http://pwp.etb.net.co/witorres/poo/3E-UML.pdf>

BOSE, Jagadeesh Chandra, 2012, *RP: Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*. PhD thesis, Eindhoven University of Technology.

BOUCKAERT, Remco R., FRANK, Eibe, HALL, Mark, KIRKBY, Richard, REUTEMANN, Peter, SEEWALD, Alex and SCUSE, David, 2013, WEKA Manual for Version 3-7-10. [online]. 2013. [Accessed 16 May 2014]. Available from: <http://ftp.jaist.ac.jp/pub/sourceforge/w/project/we/weka/documentation/3.7.x/WekaManual-3-7-10.pdf>

BOUDREAU, Tim, GLICK, Jesse, GREENE, Simeon, SPURLIN, Vaughn and WOEHR, Jack J., 2002, *NetBeans: The Definitive Guide*. O'Reilly Media, Inc. ISBN 9781449332556.

CHAKRABARTI, Soumen, COX, Earl, FRANK, Eibe, GÜTING, Ralf Hartmut, HAN, Jiawei, JIANG, Xia, KAMBER, Micheline, LIGHTSTONE, Sam S., NADEAU, Thomas P. and NEAPOLITAN, Richard E., 2008, *Data Mining: Know It All: Know It All* [online]. Morgan Kaufmann. [Accessed 15 December 2013]. Available from: <http://www.google.com/books?hl=es&lr=&id=WRqZ0QsdxKkC&oi=fnd&pg=PP1&dq=Data+Mining+Know+It+All&ots=PTEWkvTDm6&sig=Rj3trPWC0P1iCk4vluC5RailaE8>

CLAES, Jan and POELS, Geert, 2013, Process mining and the ProM framework: an exploratory survey. In : *Business Process Management Workshops* [online]. Springer. 2013. p. 187–198. [Accessed 15 May 2014]. Available from: http://link.springer.com/chapter/10.1007/978-3-642-36285-9_19

CORRAL HERRERÍAS, Antoni, 2013, Estudi de les possibles causes de l'abandonament d'un determinat pla d'estudis del departament d'Economia de la UOC. [online]. 2013. [Accessed 19 May 2014]. Available from: <http://openaccess.uoc.edu/webapps/o2/handle/10609/18955>

DE WEERDT, Jochen, DE BACKER, Manu, VANTHIENEN, Jan and BAESSENS, Bart, 2011, A critical evaluation study of model-log metrics in process discovery. In : *Business Process Management Workshops* [online]. 2011. p. 158–169. [Accessed 18 December 2013]. Available

from: http://link.springer.com/chapter/10.1007/978-3-642-20511-8_14

EDNA, Hernández Valadez and ELÉCTRICA, Ingeniería, 2006, Algoritmo de clustering basado en entropía para descubrir grupos en atributos de tipo mixto. [online]. 2006. [Accessed 16 January 2014]. Available from: <http://webserver.cs.cinvestav.mx/TesisGraduados/2006/tesisEdnaHernandez.pdf>

ESCRIBANO, Gerardo Fernández, 2006, Introducción a Extreme Programming. *Introducción a Extreme Programming* [online]. 2006. [Accessed 5 May 2014]. Available from: <http://proyectorredox.googlecode.com/svn/bibliografia%20Informe/Introduccion%20a%20Extreme%20Programming.pdf>

GAMMA, Erich and BECK, Kent, 1999, JUnit: A cook's tour. *Java Report*. 1999. Vol. 4, no. 5, p. 27–38.

HALL, Mark, FRANK, Eibe, HOLMES, Geoffrey, PFAHRINGER, Bernhard, REUTEMANN, Peter and WITTEN, Ian H., 2009, The WEKA Data Mining Software: An Update. *SIGKDD Explor. News.* November 2009. Vol. 11, no. 1, p. 10–18. DOI 10.1145/1656274.1656278.

HAN, Jiawei, KAMBER, Micheline and PEI, Jian, 2006, *Data Mining, Second Edition: Concepts and Techniques*. Morgan Kaufmann. ISBN 9780080475585.

LAKSHMANAN, Geetika and KHALAF, Rania, 2012, Leveraging process mining techniques to analyze semi-structured processes. [online]. 2012. [Accessed 24 April 2014]. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6279446

LARMAN, Craig, 1999, *UML y Patrones* [online]. Pearson. [Accessed 3 June 2014]. Available from: <http://publidisa.com/PREVIEW-LIBRO-9788483229279.pdf>

LETELIER, Patricio and PENADÉS, M^a Carmen, 2006, Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [online]. 2006. [Accessed 15 May 2014]. Available from: <http://dialnet.unirioja.es/servlet/articulo?codigo=1983605>

LY, Linh Thao, INDIONO, Conrad, MANGLER, Jürgen and RINDERLE-MA, Stefanie, 2012, Data transformation and semantic log purging for process mining. In : *Advanced Information Systems Engineering* [online]. 2012. p. 238–253. [Accessed 16 January 2014]. Available from: http://link.springer.com/chapter/10.1007/978-3-642-31095-9_16

MICHIE, Donald, SPIEGELHALTER, David J. and TAYLOR, Charles C., 1994, Machine learning, neural and statistical classification. [online]. 1994. [Accessed 18 December 2013]. Available from: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.355>

MOLINA LÓPEZ, J. M. and GARCÍA HERRERO, J., 2006, *Técnicas de análisis de datos. Aplicaciones Prácticas utilizando Microsoft Excel y WEKA*. Madrid, Universidad Carlos III.

MONTILVA, Jonás, ARAPÉ, Nelson and COLMENARES, J., 2003, Desarrollo de software basado en componentes. In : *Actas del IV. Congreso de Automatización y Control. Mérida, Venezuela* [online]. 2003. [Accessed 3 June 2014]. Available from: <http://juancol.me/rsrc/sw->

basado-en-comp-CAC2003.pdf

NAUGHTON, Patrick and SCHILDT, Herbert, 1997, *Java* [online]. McGraw-Hill. [Accessed 16 May 2014]. Available from: <http://dspace.ucbscz.edu.bo/dspace/handle/123456789/632>

PARADIGM, Visual, 2010, *Visual paradigm for uml. Visual Paradigm for UML-UML tool for software application development.* 2010.

PÉREZ-ALFONSO, Damián, YZQUIERDO-HERRERA, Raykenler and LAZO-CORTÉS, Manuel, 2012, Recommendation of Process Discovery Algorithms: a Classification Problem. [online]. 2012. [Accessed 17 January 2014]. Available from: http://www.researchgate.net/publication/256703725_process_mining_discovery_recommendation_mcpr/file/3deec523a851adcf5c.pdf

PÉREZ-ALFONSO, Damián, YZQUIERDO-HERRERA, Raykenler and LAZO-CORTÉS, Manuel, 2014, Recommendation of Process Discovery Algorithms: a Classification Problem. [online]. 17 January 2014. [Accessed 17 January 2014]. Available from: http://www.researchgate.net/publication/256703725_process_mining_discovery_recommendation_mcpr/file/3deec523a851adcf5c.pdf

PIATTINI, M. G., 1996, ODMG-93 vs SQL3: la lucha por el estándar para SGBDOO. *NOVATICA.* 1996. No. 121, p. 19–24.

ProM 6, 2013. [online], [Accessed 13 January 2014]. Available from: <http://www.promtools.org/prom6/>

RAHMAN, Ahmad Fuad Rezaur and FAIRHURST, Michael C., 2003, Multiple classifier decision combination strategies for character recognition: A review. *Document Analysis and Recognition.* 2003. Vol. 5, no. 4, p. 166–194.

RUIZ, Javier Heredia, ALMANZA, Lilián Álvarez and PONS, Naryana Linares, 2011, Comparación y tendencias entre metodologías ágiles y formales. Metodología utilizada en el Centro de Informatización para la Gestión de Entidades. *Serie Científica* [online]. 2011. Vol. 4, no. 10. [Accessed 30 April 2014]. Available from: <https://publicaciones.uci.cu/index.php/SC/article/view/484/0>

SEGRERA FRANCIA, Saddys, GARCÍA, Moreno and NAVELONGA, María, 2006, Multiclasificadores: métodos y arquitecturas. [online]. 2006. [Accessed 20 May 2014]. Available from: <http://gredos.usal.es/jspui/handle/10366/21727>

SKURICHINA, Marina and DUIN, Robert PW, 1998, Bagging for linear classifiers. *Pattern Recognition.* 1998. Vol. 31, no. 7, p. 909–930.

TÉLLEZ, A., 2005, Extracción de información con algoritmos de clasificación. *Extracción de información con algoritmos de clasificación.* 2005.

VAN DER AALST, Wil MP, VAN DONGEN, Boudewijn F., GÜNTHER, Christian W., MANS, R. S., DE MEDEIROS, AK Alves, ROZINAT, Anne, RUBIN, Vladimir, SONG, Minseok, VERBEEK, H. M. W. and WEIJTERS, AJMM, 2007, ProM 4.0: comprehensive support for real process analysis. In : [online]. Springer. p. 484–494. [Accessed 16 January 2014]. Available from: http://link.springer.com/chapter/10.1007/978-3-540-73094-1_28

VAN DONGEN, Boudewijn F., DE MEDEIROS, AK Alves and WEN, Lijie, 2009, Process mining: Overview and outlook of petri net discovery algorithms. In : *Transactions on Petri Nets and Other Models of Concurrency II* [online]. Springer. p. 225–242. [Accessed 18 December 2013]. Available from: http://link.springer.com/chapter/10.1007/978-3-642-00899-3_13

VANDEN BROUCKE, Seppe KLM, DELVAUX, Cédric, FREITAS, João, ROGOVA, Taisiia, VAN THIENEN, Jan and BAESENS, Bart, 2014, Uncovering the Relationship Between Event Log Characteristics and Process Discovery Techniques. In : *Business Process Management Workshops* [online]. Springer. 2014. p. 41–53. [Accessed 28 May 2014]. Available from: http://link.springer.com/chapter/10.1007/978-3-319-06257-0_4

WANG, Jianmin, WONG, R., DING, Jianwei, GUO, Qinlong and WEN, Lijie, 2012, Efficient Selection of Process Mining Algorithms. [online]. 2012. [Accessed 16 January 2014]. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6257371

WEBB, Geoffrey I., 2000, Multiboosting: A technique for combining boosting and wagging. *Machine learning*. 2000. Vol. 40, no. 2, p. 159–196.

WESKE, Mathias, 2012, *Business process management: concepts, languages, architectures* [online]. Springer. [Accessed 18 December 2013]. Available from: http://www.google.com/books?hl=es&lr=&id=-D5tpT5Xz8oC&oi=fnd&pg=PR4&dq=Business+Process+Management+Concepts,+Languages,+Architectures.&ots=gOmI735Eea&sig=2-SvNgQ_7f73F6XqaQHZpnxUqyY

WITTEN, Ian H. and FRANK, Eibe, 2005, *Data Mining: Practical machine learning tools and techniques* [online]. Morgan Kaufmann. [Accessed 18 December 2013]. Available from: <http://www.google.com/books?hl=es&lr=&id=QTnOcZJzIUoC&oi=fnd&pg=PR17&dq=Data+Mining+Practical+Machine+Learning+Tools+and+Techniques&ots=3goDcmXjPh&sig=AA1E7enwTJF2Tt1FtBxavVUdddY>