PRÁCTICAS 2025-2026
DESARROLLO SISTEMAS INTELIGENTES

PROYECTO:
ALERTAS DE ECG

Contenido

Objetivo

El objetivo de esta práctica es que el alumno desarrolle el prototipo de un sistema de alarmas que identifique, a partir de la interpretación del ECG, posibles patrones de riesgo para la salud.

Metodología de enseñanza

Se seguirá una metodología de enseñanza orientada a proyectos. Así, el alumno deberá desarrollar un proyecto que abarcará gran parte de los contenidos vistos en clase de teoría y llevados a la práctica en un problema concreto.

Contexto

El corazón es un órgano vital y el estudio de su morfología y comportamiento permite al médico identificar posibles enfermedades. El electrocardiógrafo hoy en día es una herramienta fundamental para estudiar el funcionamiento del corazón, de una forma eficaz, rápida y barata. Esencialmente el electrocardiógrafo registra la actividad eléctrica de los músculos del corazón, que son representados como una serie temporal e interpretados por un médico.
El procesamiento automático del ECG comienza con el procesamiento de la señal electrocardiográfica, caracterizando (identificando) las diferentes componentes morfológicas de la señal (ondas, segmentos e intervalos).
Una vez realizado este procesamiento, es el turno de desarrollar un **sistema inteligente** capaz de obtener una **conclusión diagnóstica** mediante un razonamiento automático a partir de **la fuente de datos de entrada** y **una fuente de conocimiento experto**.

Fuente de datos de entrada:

Asumiremos que existe un pre-procesamiento de la señal electrocardiográfica cuyo resultado es un fichero donde queda caracterizada de manera secuencial la señal de ECG de diferentes pacientes. A este fichero lo denominamos “log . ECG” .
Además, simplificaremos el número de derivaciones del ECG, centrándonos exclusivamente en una derivación (derivación II). En la práctica, la información que utilizaremos es sesgada y no tendrá una validez clínica real (de hecho los ficheros son “artificiales”, creados para facilitar el diagnóstico), sin embargo, desde el punto de vista del tratamiento de la información resulta bastante aproximada.
Formato de ficheros log .ECG:
El log es un fichero ASCII que contiene en cada línea la secuencia caracterizada de etiquetas del ECG de un paciente. Es decir, un paciente por línea. Las etiquetas que pueden estar presentes en cada secuencia caracterizada:

`P(<start>,<end>,<peak>)`
`Q(<start>,<end>,<peak>)`
`R(<start>,<end>,<peak>)`
`S(<start>,<end>,<peak>)`
`T(<start>,<end>,<peak>)`
`<start>=[0-9]+` comienzo de la onda en milisegundos

<end>=[0-9]+ fin de la onda en milisegundos
<peak>=[-][0-9]+.[0-9]+ pico máximo de la onda en miliVoltios.

Ejemplo:

```
P(1000,1080,0.25)
Q(1120,1220,-0.50)
R(1220,1270,1.25 )
S(1270,1370,-0.75)
T(1450,1610,0.25)
P(1910,1990,0.24)
. . .
```

Además, habrá una cabecera con algunas líneas precedidas por el símbolo '#' con información adicional sobre el ECG, en principio no utilizable por el sistema, pero útil para el ingeniero de conocimiento.

Ejemplo:

```
# Info
# ECG_Pattern : TAQUICARDIA
# Num_cycles : 50
# Heart_Rate : 145 pul/min
#
P(0,10,0.31117364322463026)
Q(15,45,-0.03882635677536976)
. . .
```

En el ejemplo se muestra que es un patrón con Taquicardia, se indica el número de ciclos que contiene el fichero y el heart rate.

Tanto el número de ciclos, como el ritmo cardíaco se pueden obtener a partir de la información del fichero que no viene precedida por un '#' (los ciclos por el número de bloques Q-T que hay, el ritmo con el número de milisegundos transcurridos del primer al último ciclo dividido por el número de ciclos), y ambas informaciones se pueden obtener mediante reglas (no en el preprocesamiento) en la inferencia basada en conocimiento. Es más adecuado hacerlo desde la base de conocimiento (también es posible desde el preproceso en Java o directamente de los metadatos precedidos por '#' pero no deberíais hacerlo así ya que aunque es técnicamente posible no es como debería desarrollarse).

Fuente de conocimiento experto:

La correcta interpretación de la señal del ECG requiere de una extensa formación y experiencia clínica. Sin embargo, parte del conocimiento esencial puede encontrarse en bibliografía y manuales de cardiología básicos. En este proyecto, la información clínica utilizada será limitada y/o simplificada para propósitos académicos y, por tanto, sin validez clínica completa.
A continuación, se enumerará un listado de las fuentes de conocimiento que deberán ser utilizadas para el desarrollo del proyecto. En primer lugar, la primera referencia (punto 1) servirá para tener una comprensión general del problema. Sin embargo, para el desarrollo del proyecto, habrá que poner especial atención al conocimiento relativo a la morfología de la señal del ECG (punto 2) y los diagnósticos que habrá que considerar (punto 3). Finalmente, se indican algunas referencias adicionales.
El uso de cualquier otra fuente de conocimiento debe ser consultada al profesor con anterioridad a su uso.

1. Electrocardiógrafo:

http://es.wikipedia.org/wiki/Electrocardiograma#Derivaciones_del_ECG

2. Morfología de la señal de ECG:

La página más moderna de Wikipedia es:

https://en.wikipedia.org/wiki/Electrocardiography

No obstante, puede ser interesante ver una versión anterior accesible a través de:

https://en.wikipedia.org/w/index.php?title=Electrocardiography&oldid=513556137#Waves_and_intervals

Con ella podéis ver los elementos morfológicos más interesantes. No todos ellos se van a aplicar o necesitar en la práctica.

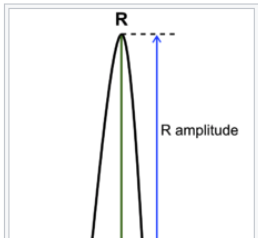
Waves and intervals

A typical ECG tracing of the cardiac cycle (heartbeat) consists of a P wave, a QRS complex, a T wave, and a U wave, which is normally visible in 50 to 75% of ECGs.[27] The baseline voltage of the electrocardiogram is known as the isoelectric line. Typically, the isoelectric line is measured as the portion of the tracing following the T wave and preceding the next P wave.

Feature	Description	Duration
RR interval	The interval between an R wave and the next R wave: Normal resting heart rate is between 60 and 100 bpm	0.6 to 1.2s
P wave	During normal atrial depolarization, the main electrical vector is directed from the SA node towards the AV node, and spreads from the right atrium to the left atrium. This turns into the P wave on the ECG.	80ms
PR interval	The PR interval is measured from the beginning of the P wave to the beginning of the QRS complex. The PR interval reflects the time the electrical impulse takes to travel from the sinus node through the AV node and entering the ventricles. The PR interval is, therefore, a good estimate of AV node function.	120 to 200ms
PR segment	The PR segment connects the P wave and the QRS complex. The impulse vector is from the AV node to the bundle of His to the bundle branches and then to the Purkinje fibers. This electrical activity does not produce a contraction directly and is merely traveling down towards the ventricles, and this shows up flat on the ECG. The PR interval is more clinically relevant.	50 to 120ms
QRS complex	The QRS complex reflects the rapid depolarization of the right and left ventricles. They have a large muscle mass compared to the atria, so the QRS complex usually has a much larger amplitude than the P-wave.	80 to 120ms
J-point	The point at which the QRS complex finishes and the ST segment begins, it is used to measure the degree of ST elevation or depression present.	N/A
ST segment	The ST segment connects the QRS complex and the T wave. The ST segment represents the period when the ventricles are depolarized. It is isoelectric.	80 to 120ms
T wave	The T wave represents the repolarization (or recovery) of the ventricles. The interval from the beginning of the QRS complex to the apex of the T wave is referred to as the absolute refractory period. The last half of the T wave is referred to as the relative refractory period (or vulnerable period).	160ms
ST interval	The ST interval is measured from the J point to the end of the T wave.	320ms
QT interval	The QT interval is measured from the beginning of the QRS complex to the end of the T wave. A prolonged QT interval is a risk factor for ventricular tachyarrhythmias and sudden death. It varies with heart rate and for clinical relevance requires a correction for this, giving the QTc.	Up to 420ms in heart rate of 60 bpm, see diagram at right for other heart rates.
U wave	The U wave is hypothesized to be caused by the repolarization of the interventricular septum. They normally have a low amplitude, and even more often completely absent. They always follow the T wave and also follow the same direction in amplitude. If they are too prominent, suspect hypokalemia, hypercalcemia or hyperthyroidism usually.[28]	
J wave	The J wave, elevated J-point or Osborn wave appears as a late delta wave following the QRS or as a small secondary R wave. It is considered pathognomonic of hypothermia or hypocalcemia.[29]	

Schematic representation of normal ECG


Animation of a normal ECG wave



Originally, four deflections were noted, but after the mathematical correction for artifacts introduced by early amplifiers, a fifth deflection was discovered. Einthoven chose the letters P, Q, R, S, and T to identify the tracing which was superimposed over the uncorrected labeled A, B, C, and D.[10]

In intracardiac electrocardiograms, such as can be acquired from pacemaker sensors, an additional wave can be seen, the H deflection, which reflects the depolarization of the bundle of His.[30] The H-V interval, in turn, is the duration from the beginning of the H deflection to the earliest onset of ventricular depolarization recorded in any lead.[31]

3. ECG y Diagnóstico:

3.1 Patrones anormales simples del ECG e hipótesis diagnósticas asociadas:
DX a considerar: Hipopotasemia, Hipocalcemia, Infarto Agudo de Miocardio temprano, Isquemia Coronaria.
Fuente:

https://en.wikipedia.org/w/index.php?title=Electrocardiography&oldid=513556137#Some_pathological_patterns_which_can_be_seen_on_the_ECG

Some pathological patterns which can be seen on the ECG

The following table mentions some pathological patterns that can be seen on electrocardiography, followed by possible causes.

Shortened QT interval	Hypercalcemia, some drugs, certain genetic abnormalities, hyperkalemia
Prolonged QT interval	Hypocalcemia, some drugs, certain genetic abnormalities
Flattened or inverted T waves	Coronary ischemia, hypokalemia, left ventricular hypertrophy, digoxin effect, some drugs
Hyperacute T waves	Possibly the first manifestation of acute myocardial infarction, where T waves become more prominent, symmetrical, and pointed
Peaked T wave, QRS wide, prolonged PR, QT short	Hyperkalemia, treat with calcium chloride, glucose and insulin or dialysis
Prominent U waves	Hypokalemia

https://en.wikipedia.org/wiki/Hypokalemia#Electrocardiogram

3.2 Patrones anormales complejos del ECG e hipótesis diagnósticas asociadas:
DX a considerar: Bradicardia Sinus, Tachycardia Sinus, Atrial Flutter. Premature Ventricular Contraction, estimating the Heart Rate.
Fuente: Bioelectromagnetism Portal. Chap. 19. The Basis of ECG Diagnosis.
http://www.bem.fi/book/19/19.htm

4. Información adicional:

Introduction to ECG Interpretation Booklet. University of Utah. 2012. Pag. 4.
[profundidad] http://ecg.utah.edu/pdf/
ECG Wave Maven. Harvard University.
[repositorio] http://ecg.bidmc.harvard.edu/

Eventos del proyecto:

1. Sesiones de seminarios.
2. Sesiones de discusión y dudas.
3. Fases del proyecto.
4. Entregables.
5. Exposiciones orales.

1. Sesiones de seminarios

En cada una de las sesiones de seminarios el **profesor explicará** las diferentes herramientas que se deberán utilizar durante el desarrollo del proyecto.

Además, podrá haber un conjunto de ejercicios que se le puedan presentar al alumno para practicar con la herramienta correspondiente y poder así identificar posibles problemas y facilitar el aprendizaje de esta.

Los seminarios que se realizarán serán los siguientes:

1. Seminario de desarrollo de ontologías con *Protégé* (1 sesión).
2. Seminario de desarrollo en JAVA-*DROOLS* (4 sesiones).
3. Seminario de fuzzy tools (1 sesión).

2. Sesiones de discusión y dudas

Las sesiones de discusión tienen por objetivo poder discutir acerca del diseño o posibles problemas en cada una de las fases del proyecto. El **alumno**, en este caso, será el **elemento activo** y será quien lleve a clase aquellas cuestiones que considere.

3. Fases del proyecto

El proyecto se divide en las siguientes subfases.

Fase 1.A. Desarrollo de la ontología del dominio (obligatorio).
Fase 1.B. SI básico (obligatorio)
Fase 2. SI completo (obligatorio)
Fase 3. SI extendido (opcional)

4. Entregable final y entregable voluntario

El proyecto tiene un **único entregable obligatorio**, denominado **entregable final**, donde se hará un compendio con los diferentes informes que el alumno habrá elaborado durante el proyecto.
Este entregable incluye **todo lo desarrollado en las Fases 1,2 y 3** y, además, si se ha **desarrollado la Fase 3, también se incluirá en esta entrega**.
No obstante, con el fin de que el alumno pueda saber cuál es el estado del desarrollo del proyecto antes de que este finalice, habrá la posibilidad de hacer un entregable parcial y una presentación oral del mismo. Este entregable no será evaluado ni supone puntos extras.

Nombres de proyectos y nombres de ficheros.

Para facilitar la identificación normalizaremos los nombres de los proyectos (y directorios) y de los ficheros. Para el proyecto Protégé de la ontología el proyecto debe ir en un directorio que se llame:
DSI_ONT_[OPC|FINAL|EXTRA]_<Nombre_de_grupo>
En el nombre de grupo no uséis símbolos especiales (tildes, eñes, etc.) y cambiad los espacios por underscores "_". Para la entrega opcional usad la opción "_OPC_" y para la entrega final "_FINAL_". Es decir, que si el grupo se llama "Informática Forever" el proyecto de ontología para la entrega opcional iría en un directorio que se llamaría DSI_ONT_OPC_Informatica_Forever, y la ontología del proyecto final

DSI_ONT_FINAL_Informatica_Forever. Para el nombre del proyecto dentro del propio programa Protégé reusadlo.
Para el proyecto Eclipse, parecido, llamad al proyecto en el propio eclipse:
`DSI_KS_[OPC|FINAL|EXTRA]_≤Nombre_de_grupo>`
Eso os creará directamente un directorio con ese nombre, que es lo que debéis luego comprimir y enviar.

Entregable voluntario y final

El entregable final se enviará a través del Aula Virtual (ver fechas de entrega en planificación temporal) y constará de una serie de informes y código fuente.
En concreto el entregable final constará **de UN fichero ZIP** que, a su vez, contendrá 2 o 3 zip conteniendo el trabajo realizado en cada fase.
El nombre de ese único fichero zip será:
DSI _[OPC|FINAL]_<Nombre_de_grupo>.zip
y, en su interior, contendrá los siguientes .zip que contienen:

- Fase1A.zip. Contendrá
  - el directorio con el proyecto Protégé.
  - Informe_1.pdf Informe del diseño de la ontología.
- Fase1B.zip. Contendrá
  - el directorio con el proyecto Eclipse con Sistema Inteligente básico.
  - Informe_2.pdf Informe sobre el SI básico.
- Fase2.zip. Contendrá
  - el directorio con el proyecto Protégé actualizado (con cambios conforme al desarrollo extendido).
  - Informe_1.pdf Informe del diseño de la ontología FINAL (con cambios conforme al desarrollo extendido).
  - el proyecto Eclipse con SI completo.
  - Informe_3.pdf Informe sobre SI completo.
- *Fase3.zip. (opcional) Contendrá*
  - *el directorio con el proyecto Protégé actualizado (con cambios conforme al desarrollo extra).*
  - *Informe_1.pdf Informe del diseño de la ontología FINAL (con cambios conforme al desarrollo extra).*
  - *el directorio con el proyecto Eclipse con extensiones.*
  - *Informe_4.pdf (opcional). Informe sobre extensiones del SI.*
- Exposición oral.

A continuación, se explican cada una de estos informes y fases.

Informe 1. Desarrollo de la ontología del dominio siguiendo alguna metodología. La ontología deberá reflejar los conceptos, atributos y relaciones fundamentales que se plantea para la descripción del dominio clínico, así como sus relaciones y construcción básica.
La ontología debe centrarse en aquellos conceptos que podrán ser utilizados durante el desarrollo del sistema. Por ejemplo, en este enunciado se habla de Ondas, Complejos e Intervalos, por tanto, deben aparecer en la ontología. Sin embargo, en las fuentes de conocimiento se mencionan otros muchos aspectos que quedarán fuera del alcance de este proyecto (por ejemplo, los electrodos).
La ontología debe ser lo más completa posible y **debe incluir**: conceptos, relaciones, propiedades, metadatos (para todos los conceptos y relaciones) e

instancias de cada concepto/relación no abstracto (**al menos una** por cada concepto y relación).
Fase1A.zip Contendrá: El directorio de un proyecto en *Protégé* (ficheros) y el informe_1.PDF (máx. 4 páginas) explicando las principales decisiones del diseño.

Informe 2. Consiste en desarrollar un SI capaz de leer un fichero de log de ECG con un motor de reglas **que, como mínimo sea capaz de calcular los diferentes intervalos y complejos de un ECG, número de ciclos y la frecuencia cardíaca**. Se aconseja que para esta entrega opcional también sea capaz de mostrar por pantalla (no es necesario generar la salida a fichero ni modificar la base de hechos, puede ser un printf en el consecuente de la/s regla/s que lo detecta) la detección de Taquicardia y Bradicardia.
Este informe 2 es el resultado de finalizar la Fase 1B.
Fase1B.zip Contendrá: el directorio con el proyecto *Java-Eclipse* (**con codificación UTF-8**) y la memoria PDF (máx. 4 páginas) explicando las principales decisiones del diseño.

La entrega opcional sería un fichero llamado DSI_OPC_<Nombre_del_grupo>.zip que incluiría el fichero Fase1A.zip+ Fase1B.zip+Video de presentación 5-10 min (voluntario)

Informe 3. Consiste en desarrollar un motor completo que además de tener la funcionalidad anterior (Fase 1B) sea capaz de realizar **diagnósticos** de:

- taquicardia,
- bradicardia,
- hipocalcemia,
- hipopotasemia,
- contracción ventricular prematura,
- isquemia coronaria,
- e Infarto Agudo de Miocardio.

Este informe 3 es el resultado de finalizar la Fase 2.
Fase2.zip Contendrá: el directorio del proyecto *Java-Eclipse* (**con codificación UTF-8**), y la memoria (máx. 10 páginas, en **PDF**) explicando las principales características del sistema y una breve explicación (a modo de manual de usuario) de cómo simular una ejecución completa.

Informe 4. Es opcional, consiste en desarrollar motor extendiendo el SI de la Fase 2, soportando todos los mecanismos de razonamiento anterior, pero con nuevas funcionalidades. La validez de estas funcionalidades nuevas debe ser previamente consultadas al profesor.
Estas mejoras tendrán que ser significativas y creativas. Pueden suponer mejoras en el diagnóstico o funcionalidades. Esencialmente deben suponer mejoras en la base de conocimiento (reglas y metadatos), su ejecución y su consulta. Repetición de los desarrollos ya realizados con leves modificaciones no se considerarán mejoras.
Otras mejoras referentes a aspectos de programación tradicional no serán valoradas.
Este entregable extra es el resultado de finalizar la Fase Extra, que es opcional.
Fase3.zip Contendrá: un proyecto Eclipse-Java (**con codificación UTF-8**) y la memoria (máx. 5 páginas en **PDF**) explicando las principales características de mejora respecto la Fase 2.

La entrega FINAL OBLIGATORIA sería un fichero llamado DSI_FINAL_<Nombre_del_grupo>.zip incluirá el fichero Fase1A.zip (con las modificaciones posibles a la Ontología que pudieran surgir en la Fase 2) + Fase2.zip + Fase3.zip

Entregable Voluntario

Como se ha comentado mitad del desarrollo del proyecto, de forma opcional se podrá presentar lo desarrollado en Fase1A.zip y Fase1B.zip con el fin de obtener **feedback**. Además, la presentación de la entrega es voluntaria también, y si se desea se puede hacer mediante un video de unos 5-10 min que se adjunta a la entrega o si se quiere puede hacerse de forma presencial, pero vuelvo a recordar que esto es completamente opcional.
Esta entrega no tendrá ningún efecto directo en la calificación, pero es recomendable porque así no iréis a ciegas a la entrega final.

5. Presentaciones orales

Presentación oral obligatoria
Habrá una presentación oral obligatoria al final del proyecto. Algunos miembros de cada equipo, elegidos por sorteo como representantes, expondrán brevemente el trabajo realizado y el porqué de las decisiones tomadas. El profesor le realizará preguntas indagatorias o le invitará a clarificar elementos del trabajo y la exposición.
La duración de las presentaciones dependerá del número de equipos y disponibilidad horaria.
Por lo general la exposición constara de 20 min donde el alumno dispone de hasta 15min de exposición y el profesor tendrá 5 min de preguntas.
Esta presentación se evaluará y formará parte de la calificación de todos los miembros del equipo. Es importante señalar que la nota de prácticas podría ser diferente para cada miembro del grupo, incluida la posibilidad de que algunos aprueben y otros suspendan dentro del mismo grupo.

Presentación oral voluntaria
Aquellos alumnos que hayan realizado el entregable voluntario podrán, si así lo piden, hacer una presentación oral exponiendo el trabajo realizado en las Fases 1A y 1B.
Si se desea se puede hacer mediante un video de unos 5-10 min que se adjunta a la entrega en vez de presencial.
Esta presentación **no tendrá efecto en la calificación**.

6. El equipo y el vocal

Los equipos se constituirán por 3 miembros en función de criterios de idoneidad y afinidad. En casos muy excepcionales y justificados podrían hacerse equipos de 2 personas (p.e. para cuando el número de alumnos total no es divisible por 3). No podrá haber equipos formados por un alumno o más de 3 alumnos.
Los grupos pueden tener miembros de diferentes subgrupos pero deben tener en cuenta que las presentaciones son en horario de prácticas, y que en un grupo con miembros de diferentes subgrupos algunos podrían tener problemas de colisión con

otras asignaturas. En su momento se ofrecerán en la herramienta "Apúntate" del aula virtual diferentes slots dentro de los horarios de prácticas para escoger y realizar la entrevista. Será responsabilidad del vocal/portavoz del grupo escoger un slot en el que todos sus miembros puedan acudir. Solo aceptaré las solicitudes en Apúntate realizadas por los vocales/portavoces, las del resto de alumnos serán anuladas. Igualmente, debo indicar que en caso de que slots de un subgrupo se acaben, y se quede algún grupo de alumnos con la totalidad de sus miembros pertenecientes a ese subgrupo, que tendrán PRIORIDAD sobre grupos con alumnos de varios subgrupos. Es decir, que en caso de no haber huecos hay una prioridad basada en el número de alumnos de ese subgrupo (primero los grupos con los 3 alumnos de ese subgrupo, luego los de 2 alumnos, y finalmente los de 1… dentro de la misma prioridad el criterio es el primero que lo pide lo recibe).
En resumen, el vocal/portavoz será el responsable de:

1. coordinar la comunicación entre el profesor y los miembros del equipo,
2. informar a sus compañeros de las comunicaciones/solicitudes que le haya hecho el profesor
3. realizar las entregas de cada fase por el aula virtual (solo las puede hacer el vocal)
4. pedir el slot para las presentaciones obligatorias (solo la puede pedir el vocal)

7. El sistema basado en conocimiento

El programa Java (Fase 1, 2 y 3) debe ejecutarse teniendo como parámetros un directorio de entrada, y un directorio de salida. Del directorio de entrada leerá todos los ficheros <nombre fichero>.ecg que encuentre y aplicará, uno a uno, la inferencia de diagnóstico. En el directorio de salida escribirá tanto un ficheros de salida individuales <nombre fichero>.salida.txt por cada fichero <nombre fichero>..ecg que haya leído en el directorio de entrada, como un fichero todo.salida.txt donde aparecen el nombre de cada fichero leído y, a continuación su diagnóstico correspondiente. Este último fichero, todo.salida.txt, también debería mostrarse por la consola de Java antes de finalizar la ejecución (tan sencillo como añadir unas instrucciones al final del programa para que lea el fichero y lo muestre por pantalla).

El **programa debe permitir una ejecución sin parámetros**, creando dos variables en el programa Java (`String dirEnt`, y `String dirSal`) que se deben inicializar al valor donde está el directorio con los ficheros .ecg, y al directorio donde se escribirán los ficheros de salida. Es decir, el programa tiene esas dos variables, si se detecta que se ha ejecutado con 1 argumento, el valor de dicho argumento reescribe el valor de `dirEnt`, y si se ha ejecutado con 2 argumentos, reescriben `dirEnt` y `dirSal` respectivamente.

Modo de entrega

Recordemos que el entregable opcional se entregará del siguiente modo:

- Adjuntar el fichero DSI_OPC_<Nombre_equipo>.zip en el **Aula Vitual** en la Tareas que se destine al efecto. El vocal es el encargado de subirlos.
  El .**ZIP** contendrá
  - Obligatoriamente: Fase1A.zip, Fase1B.zip (cada uno con fuentes y su pdf correspondiente).

- **Muy importante**. El nombre del PROYECTO JAVA-ECLIPSE (en Fase_1B.zip) debe ser:
    - DSI_KS_OPC__<Nombre_equipo>.

El entregable final se entregará del siguiente modo:

- Adjuntar el fichero DSI_FINAL_<Nombre_equipo>_DSINT.zip en el **Aula Vitual** en la Tarea que se destine al efecto. El vocal es el encargado de subirlos. El .**ZIP** contendrá
    - Obligatoriamente: Fase1A.zip, Fase2.zip (cada uno con fuentes y su pdf correspondiente).
    - Opcionalmente: Fase3.zip
- **Muy importante**. El nombre del PROYECTO JAVA-ECLIPSE (en Fase2.zip) debe ser:
    - DSI_KS_FINAL__<Nombre_equipo>.
- **Muy importante**. El nombre del PROYECTO JAVA-ECLIPSE (en Fase3.zip) debe ser:
    - DSI_KS_EXTRA__<Nombre_equipo>.

Es decir, **para la entrega final no es necesario entregar la FASE 1B**, ya que esta es un paso intermedio de la FASE 2. Aquellos que habéis hecho una entrega opcional debéis modificar la FASE 1A para ser coherente con la FASE 2.

Fase1A.zip Contendrá: El proyecto en Protégé (ficheros) y el informe1.PDF (máx. 4 páginas) explicando las principales decisiones del diseño de la ontología.

Fase2.zip Contendrá: el proyecto Java-Eclipse (codificación UTF-8 y su nombre correcto DSI_KS_FINAL__<Nombre_equipo>) y la memoria (máx. 10 páginas en PDF) explicando:

a) las principales decisiones del diseño
b) las principales características del sistema
c) una breve explicación (a modo de manual de usuario) de cómo simular una ejecución completa.

Fijaos que el punto (a) es lo que se pedía en el informe de la FASE 1B en el documento de prácticas, y el (b) y (c) lo que se pedía para la FASE 2 en ese mismo documento.

IMPORTANTE: Comprimir SOLAMENTE en formato .ZIP. Usad codificación UTF-8 en el proyecto Eclipse. Poned el nombre del proyecto Eclipse que se especifica para cada Fase/entrega y, para los documentos/memoria, usad PDF.

¡MUY IMPORTANTE! No se considerarán entregadas correctamente aquellas prácticas que no sigan estas indicaciones y pueden acarrear directamente el suspenderlas.

Evaluación

- Las prácticas deberán ser superadas (5 puntos) para poder hacer media con la parte de teoría.
- Es obligatorio realizar y presentar oralmente el trabajo de la Fase 1 y Fase 2 para superar las prácticas.
- La Fase 3 únicamente se podrá realizar y presentar si se han realizado también las fases anteriores. Es decir, no se evaluarán mejoras sujetas a una de las fases si no se han completado todas.
- **La presentación oral de todo el proyecto es obligatoria e imprescindible para poder superar las prácticas.**
- **La presentación oral se hará por videoconferencia aunque es posible hacerla presencial si el profesor así lo indicase. El profesor tiene la última palabra a la hora de optar o no por hacer la presentación presencial.**

Evidencias para la evaluación

En general, la evaluación de las prácticas tendrá como evidencias de evaluación:

1. El entregable enviado y el informe de seguimiento.
2. La exposición oral, donde los miembros que exponen representan a todos los integrantes, compartiendo la misma evaluación.
3. La entrevista final donde se citará a cada uno de los miembros del equipo a aclarar dudas.

La evaluación por fases (definitiva):

FASES:	**Ptos.**
Fase 1A: Ontología (obligatorio) e informe 1	**2**
Fase 1B: Elementos pedidos para el S.I. básico (obligatorio) e informe 2	**2**
Fase 2: Elementos pedidos para el *S.I. completo* (obligatorio) e informe 3	**4**
Fase 3 (extra): SI extendido (opcional) e informe 4	**2**
Presentación (obligatorio)	**1**

Puntos máximos obtenibles: **11 puntos (nota máxima en acta: 10)**.
Entregables fuera de plazo:
Un entregable parcial fuera de plazo puede enviarse hasta con dos días de retraso (perdiendo los 0,5 puntos por día de retraso). Después de esos **dos días no se aceptarán** entregables , teniendo una calificación de SUSPENSO en las prácticas de la asignatura.

Es importante indicar que en la presentación se ofrecerá una nota preliminar pero que siempre queda sujeta a modificación por los resultados de la presentación y ajuste de la nota de las diferentes fases conforme a lo aclarado en la presentación (tanto para subir como para bajar). Igualmente se puede considerar que algún miembro del grupo no justifica su participación o aportación en el trabajo entregado y se le puede poner una nota diferente del resto del grupo.

Criterios generales de evaluación:

- Tanto en cuestiones de diseño como de desarrollo se considerará tanto la corrección como la completitud de la solución propuesta.
- Se otorgará gran valor a las explicaciones dadas en los informes. Éstos deberán justificar de forma clara el por qué de los diseños realizados.
- **La detección de cualquier tipo de plagio se penalizará para todos los miembros de los grupos implicados (sin entrar a valorar quien copia a quien) con la calificación de *SUSPENSO* en la convocatoria que proceda**. Se podrá aplicar la normativa de la Universidad de Murcia al respecto sobre autoría y plagio aplicándose las medidas disciplinarias ahí reflejadas.

Notas generales sobre la práctica

Nota: Es normal si no entendéis algunas de estas notas antes de poneros a hacer la práctica. Lo que aquí se menciona se podrá entender a medida que trabajéis en la práctica.

A. En las documentaciones/entrega:
   1) **Poned el nombre del grupo.**
   2) **Poned los nombres de los miembros del grupo.**
   3) **Poned los subgrupos de cada miembro.**
   4) **Usad codificación UTF-8 en ficheros de proyectos.**

   No hacerlo conlleva penalización en la nota.

B. Es importante que TODAS las clases concretas de la ontología tengan UNA instancia ejemplo/prueba.
C. Es importante que TODAS las clases tengan algo en Documentation (meta-datos), y si ponéis la referencia de donde sacasteis esa información (p.e. el link), mejor.
D. Comentad si seguís alguna metodología para hacer la ontología, haciendo apartados con sus fases, los "resultados" de estas, y las justificaciones de las decisiones tomadas.
E. Podríais incluir grafos y diagramas con la ontología, o bien hechos a mano y escaneados, o bien usando algunos de los plugins que ofrece protegé y aparecen en las transparencias, p.e. Jambalaya, TgVizTab
   (Para activar las pestañas de los plugins id a Project->Configure y en la pestaña "Tab Widgets" activar los plugins que queráis, p.e. Jambalaya)
   P.e. con Jambalaya podéis probar p.e. el QuickView de "Class&Instance Tree" y luego probar el Radial o Spring Layout (iconos bajo la pestaña Jambalaya).
   Esos gráficos los podéis grabar (p.e. Menu->Jambalaya->File->Export...)
   Otro ejemplo, con TgVizTab podéis seleccionar una clase en el class browser, pinchar en add-class y luego en crear gráfico (en la ventana

settings, justo encima de la class browser)... podéis jugar con el "radio" etc.

F. El documento 1 debería incluir cierta explicación del proceso por el que tomáis las decisiones de diseño que tomáis, y su **justificación**.
G. Recordad incluir **RESTRICCIONES** en vuestra Ontología (3 como mínimo).
H. No debéis programar siguiendo el paradigma imperativo/orientado a objetos... (p.e. cojo un hecho "contenedor" de muchos hechos e itero calculando esto, o lo otro... o cojo un objeto que contiene la información y luego extraigo uno de sus componentes para hacer esto o lo otro). Con esto nos referimos a que, por ejemplo, los consecuentes de las reglas no deberían ser "programas" o trozos de algoritmo (p.e. no debería ser un bucle for) si no más bien limitarse a realizar solo inserciones, modificaciones, borrados de un hecho, etc… En general, en el consecuente no debería haber más código java que inserciones/modificaciones/eliminaciones de hechos y quizá mostrar algo por pantalla (para debug)… no mucho más.
   Además, los antecedentes deberían ir directamente a los conceptos que se van a usar: p.e.

   Si se entiende que un segmento PQ, es una P y Q del mismo ciclo, pues: en vez de:

```
when
  $c: Ciclo()
then
  insert(new SPQ($c.p.inicio, $c.q.fin,$c.ciclo));
```

   sería mejor:

```
when
 $p: P()
 $q: Q(ciclo == $p.ciclo)
then
 insert(new sPQ($p.inicio,$q.fin,$p.ciclo));
```

   donde en el consecuente ejecutamos un "extraer" de la P y la Q de un hecho Ciclo que es un "contenedor".

   Ambas funcionan pero en la segunda versión se ve, en el antecedente, qué constituye un segmento PQ (y el consecuente solo lo inserta), mientras que en la primera se usa la regla para localizar un objeto (el ciclo) sobre el que trabajar (y poner en el consecuente el trabajo a realizar)... la primera regla es un bucle `for` encubierto, se itera por todos los ciclos existentes.
I. La forma más "correcta" de mostrar que se ha hallado un diagnóstico no es imprimirlo en la regla que lo "descubre" (quizá es interesante en la fase de debug) sino que, en principio, se deberían insertar los "hechos" de diagnósticos y que luego hubieran reglas que hicieran los "reports" finales (posiblemente en una agenda-group de "report").
J. Puede parecer muy engorroso hacer una clase que ni siquiera tiene nuevos atributos, o clases abstractas que aparentemente tampoco, pero es solo un efecto secundario indeseado del hecho de que drools está

construido con java, y son necesarias las clases para crear hechos... pero en este tipo de paradigma de programación son las ocurrencias de los conceptos más concretos de la jerarquía (instancias de las clases más profundas) las que mueven la inferencia de manera más natural.
No significa que no haya reglas sobre conceptos más generales, por supuesto que las hay, pero las reglas deben trabajar al nivel de abstracción sobre el que se refiera la condición, y no sobre niveles más abstractos y añadir como condición extra indexaciones indirectas y encadenadas a aspectos más profundos, organizados así por conveniencia de programación o costumbre de OO-imperativo (p.e. un objeto que contiene a otros a los que se accede indirectamente... p.e. un objeto ciclo, que contiene ondas y que, para preguntar sobre si una onda tiene determinadas características pues se hace una regla que busque un ciclo que contenga una onda que cumpla tal, en vez de buscar una onda que cumpla tal... que es su nivel real de abstracción, el de onda, no el de ciclo).

K. No se deben ver las clases como en orientado a objetos, es decir, un tipo abstracto de datos (colección de información y enlaces a otros objetos), junto con elementos funcionales (métodos) y capacidad de herencia, sino que hay que ver las ontologías como una estructura de conceptos que contienen información que los define y una estructura taxonómica... no diseñéis las clases para programar con ellas ni para facilitar dicha programación sino para representar hechos (con los que luego jugarán las reglas).
L. Insisto, **no debéis "programar" en las reglas**, el consecuente debe ser básicamente un insertar/modificar/eliminar instancias. Los antecedentes, deben ser patrones de hechos lo más simples posibles.
    - Si tus consecuentes parecen un mini programa, vas por mal camino.
    - Si tus antecedentes tienen patrones con valores que necesitan "navegar" por varias referencias indirectas (navegar por punteros), vas por mal camino (no estás al nivel correcto de abstracción).
M. Cuidado con la tentación de programar siguiendo el paradigma de Orientado a Objetos. Se tiene la tentación de "optimizar" a base de crear objetos contenedores de la información, a la que accedéis "navegando" por varias referencias indirectas (navegar por punteros) pero un sistema basado en reglas funciona de otra manera... se busca simplicidad y facilidad de entender las reglas, así que es al contrario, la base de puebla de hechos básicos que las reglas puedan acceder directamente lo que las hace mucho más legibles incluso para profanos de la programación (y esa es la idea). Todo el sistema de inferencia está optimizado para precisamente ese tipo de estructuras "desempaquetadas".
N. Muchos tendréis la tentación de trabajar con la clase abstracta Onda, cuando deberíais usar instancias de las clases más concretas… p.e. P() o Q() en vez de Onda(tipo=="P")... lo primero (usar P() o Q()) es su nivel correcto de abstracción, lo segundo es programación a un nivel superior de abstracción por cuestiones de programación con el paradigma OO-Imperativo.
Las cosas se os simplificarían mucho si en las ondas ponéis información sobre su ciclo... ya ni necesitaríais objetos de tipo ciclo.

Veréis que, con todas las ondas, segmentos, etc. presentes como hechos (instancias) a ese nivel las reglas de diagnóstico (y creación de segmentos, etc) son sencillísimas y muy fáciles de interpretar.

O. Aunque sea algo repetitivo.
Reglas del tipo:

```
when
  $c: Ciclo()
then
  ...
```

es un bucle... sospechad que posiblemente estéis "programando" algorítmicamente... especialmente si accedéis a atributos de $c, o indexáis a través de ellos... preguntaos si se puede hacer a un nivel más bajo y que los antecedentes contengan más restricciones (sean más concretos).
De nuevo esto no implica que no puedan, ni deban, existir reglas generales (que usen elementos abstractos o menos restricciones), pero estad atentos a ellas.

P. Es normal que puedan aparecer repeticiones de un mismo diagnóstico (y varias instancias del diagnóstico concreto con diferentes ciclos donde se haya producido). En principio deberíais mostrar solo uno de la misma enfermedad. Es fácil crear una regla que actúe sobre la clase abstracta Enfermedad (o Diagnóstico, o como hayáis llamado a la clase genérica) y escoger solo a la instancia de menor ciclo si hubiere repetidas. P.e.

```
when
  $e: Enfermedad()
  not (Enfermedad(getClass()==$e.getClass(),
                  ciclo<$e.ciclo))
then
...
```

Recordad, además, que debería aparecer, cuando listéis los Diagnósticos o Enfermedades, información sobre la "razón" por las que se ha hecho dicho diagnóstico. P.e.:

```
  Hipocalcemia debido a QRST muy largo en ciclo 13
```

Q. No hay problema en que aparezcan en uno de los ficheros más enfermedades que las que pone en el nombre del fichero. Lo que SI ES UN PROBLEMA es si NO os aparece una de las enfermedades del nombre del fichero, u os aparecen enfermedades en el fichero "normal".

R. Se pueden entrar en bucles infinitos si tu consecuente (RHS) modifica algo de tu mismo antecedente (LHS), lo que provoca reactivarse... para eso están los atributos no-loop, y lock-on-active.
¿Cómo van? pues gracias al maravilloso stackoverflow (no me atribuiré autoría de tan excelente explicación en https://stackoverflow.com/questions/17042437/what-is-the-difference-between-no-loop-and-lock-on-active-in-drools)
Long story short:
no-loop: avoid the re-activation of a rule caused by the RHS of that SAME rule.

lock-on-active: avoid the re-activation of a rule NO MATTER what the cause is.
Long story: http://ilesteban.wordpress.com/2012/11/16/about-drools-and-infinite-execution-loops/

S. Es importante que entendáis que en el programa Java desde el que se usa Drools solo debéis hacer labores de procesamiento de datos, pero no de inferencia. Así, en esta práctica podéis hacer en Java un parser del fichero de datos ECG, para obtener el heart rate, las ondas P, Q, etc., pero no para obtener los complejos, segmentos, etc.. Es decir, en Java se pueden obtener los hechos básicos directamente extraíbles de los datos de entrada e insertarlos en la base de conocimiento, pero no para inferir hechos nuevos o conceptos complejos no directamente señalados en los datos e insertarlos.

Agendas y su uso en el SI

Para el Sistema Inteligente se deberían usar agendas (se verán en los seminarios de Drools). Una forma básica de usarlas y cumplir este requisito podría ser crear algunas reglas que realicen la obtención de conceptos complejos a partir de los básicos (la entrada son ondas individuales). Una vez obtenidos los conceptos complejos se puede tener otra agenda para hacer el diagnóstico en sí.

Uso de la IA:

Todos los recursos y materiales externos que se utilicen en los ejercicios evaluables —incluyendo herramientas de Inteligencia Artificial, recursos en línea, libros y artículos— deberán ser citados adecuadamente tanto en el código fuente como en la documentación.

Cualquier otra duda o explicación

Podéis realizar preguntas o dudas en las sesiones de trabajo, o vía tutorías. Tratad de no abusar, en el sentido de preguntar cada mínima decisión que toméis. Se os da libertad creativa con todas las consecuencias (es decir, sabiendo que haréis todo tipo diferente de cosas, sin que unas sean inherentemente las únicas buenas) y sobre un dominio abierto a muchas variantes. Las decisiones que repercuten en la nota negativamente son solo aquellas que surgen de aplicar claramente mal los conceptos de la asignatura y si los sistemas no son capaces de hacer los mínimos que se piden. No olvidéis **justificar** vuestras decisiones, posiblemente más importante que “hacer”, es **justificar** lo que se hace.