

Ingeniería del Software II - Enunciado de la práctica para el curso 2021/2022

Enunciado: Gestión de pacientes con enfermedades infectocontagiosas

Teniendo en cuenta la reciente experiencia con la pandemia por COVID 19, el SESCAM nos ha pedido realizar un software para la gestión de pacientes con enfermedades infectocontagiosas. Los diferentes módulos de la aplicación serán utilizados por diferentes actores dentro del SESCAM, pero no todos los actores podrán utilizar todos los módulos, por lo que será necesario **restringir adecuadamente el acceso a la ejecución**.

Catalogación y mantenimiento de enfermedades infectocontagiosas y generación de campañas informativas

Una de las principales preocupaciones del SESCAM es **mantener actualizado una base de datos con todas las enfermedades infectocontagiosas**. Para ellos es importante mantener no sólo la lista de enfermedades sino también de los posibles síntomas que se pueden padecer, de la temporalidad de las enfermedades, de las posibles medicinas aprobadas por los Organismos Sanitarios a nivel regional y nacional, posibilidad de confinamientos, y dado el caso, de las posibles vacunas y campañas de vacunación a personas con distintos niveles de vulnerabilidad.

Además, a fin de poder mantener convenientemente informada a la ciudadanía, se desarrollarán campañas informativas preventivas, aunque si bien el objetivo de la herramienta no es dar soporte a las campañas informativas, sí lo es llevar un **inventariado y catalogación de las campañas realizadas**, así como su coste, temporalidad y su efectividad, de modo que se puedan realizar estudios sobre el impacto de las enfermedades infectocontagiosas en la población castellano-manchega y en el tejido empresarial y económico.

Detección temprana y **seguimiento de positivos en las diferentes enfermedades**

A fin de poder disponer por anticipado de datos suficientes sobre las posibles epidemias /pandemias para adquirir medicinas y/o vacunas por anticipado es necesario poder realizar una detección temprana de los contagiados. En este sentido, una persona puede estar en uno de los siguientes estados:

- Enfermo: si está actualmente o estuvo infectado con alguna de las enfermedades recogidas en la base de datos. Debe ser posible distinguir si no necesita confinamiento, o si necesita atención hospitalaria en distintos niveles (por ejemplo, no aislamiento, aislamiento).
- Sano: si no está infectado con ninguno de los patógenos identificados en la base de datos
- Sano, pero en cuarentena: si está sano, pero ha estado en contacto con alguno de los enfermos por alguna de las enfermedades infectocontagiosas.
- Enfermo en cuarentena: si estando infectado por alguna de las enfermedades recogidas en las bases de datos, necesita cuarentena y cuánto tiempo.

En caso de que una persona esté enferma, o esté en cuarentena por contacto con un sospechoso de enfermedad, es necesario registrar tanto su estado como la red de contactos que ha tenido para poder alertar a sus allegados convenientemente.

Generación y ejecución de plan de vacunaciones para cada una de las enfermedades

En caso de aviso por una epidemia o pandemia de alguna de las enfermedades infectocontagiosas, se lanzarán campañas de vacunación.

Para realizar un calendario de vacunación, debe registrarse también las previsiones de aprovisionamiento de vacunas. Este plan de vacunación debe cubrir las pautas de vacunación recomendadas, incluyendo la aplicación de las diferentes dosis de acuerdo con las pautas establecidas por la organización mundial de la salud.

La ejecución de este plan de vacunación requiere acceso a la información personal de la ciudadanía para establecer calendarios de vacunación. Se empezará informando por SMS a aquellos pacientes a los que se consideren que tiene un nivel de vulnerabilidad mayor.

Realización de estadísticas

Los directivos del SESCOAM están interesados en la realización de determinadas estadísticas, tales como:

1. Número de enfermedades y número de personas en cada uno de los estados indicados anteriormente.
2. Número de enfermedades catalogadas
3. Número de campañas informativas realizadas por cada una de las enfermedades catalogadas
4. Evolución de las enfermedades a lo largo del tiempo.
5. Estadísticas de vacunación por cada una de las enfermedades
6. Relación entre personas vacunadas y personas infectadas en un periodo de tiempo.
7. Previsión de olas de contagio por cada una de las enfermedades por intervalos de tiempo, teniendo en cuenta la evolución de las enfermedades.

Trabajo a realizar

El objetivo principal de los laboratorios **es el desarrollo del proyecto anteriormente descrito utilizando los procesos software que estamos viendo en teoría**. Por tanto, tendréis que abordar diferentes decisiones de diseño que condicionarán el coste final del proyecto, siempre teniendo en cuenta las limitaciones que consideréis, y usando las herramientas de control, configuración y comunicación que habéis establecido. No obstante, si por alguna razón veis que los recursos humanos y materiales que habéis establecido no son suficientes, plantead la posibilidad de incorporar nuevos recursos.

Como metodología de desarrollo se utilizará el **Proceso Unificado de Desarrollo (PUD)** que se ha explicado en clase. Haga todas las suposiciones que considere necesarias y adecuadas, pero explicándolas y razonándolas. Las tecnologías de desarrollo dependerán de vuestra elección; en cualquier caso, la opción preferida es el desarrollo en **Java 8**.

A continuación, se indica lo que se espera de los equipos de trabajo para cada uno de los bloques, teniendo en cuenta que el ejercicio se tiene que ver como un todo.

Bloque 1

Tema 1 - Planificación y ejecución del proyecto con el Proceso Unificado de Desarrollo (PUD).

Cuando se realiza la planificación del proyecto con el PUD, los parámetros que se deben obtener son el **coste final** del proyecto, la **agenda** del proyecto y la **fecha de finalización** (lo que debe incluir un **calendario de liberaciones o releases**).

Por simplificar, haremos las siguientes suposiciones:

- Un Requisito Funcional se mapeará en un solo Caso de Uso (1:1),
- Un Caso de Uso se realizará en una y solo una iteración (1:1).
- Por simplicidad asumiremos que tendremos una *Iteración 0* en la fase de inicio, varias iteraciones correspondientes a la realización de los diferentes casos de uso, y una última *Iteración N*, que se ejecutaría en la fase de transición y donde se realizarían tareas relacionadas con el cierre del proyecto, como la integración, las pruebas de integración, el despliegue, el manual de usuario, etcétera. Supongamos que el coste de la iteración 0 es de 1000 € (sería muy interesante estimar el coste de esta iteración realmente, pero por simplicidad asumiremos ese coste), y el coste de la iteración N es de 2000 €. Tenga en cuenta que esos costes son los correspondientes a recursos humanos. Discuta si hay algún coste más que se deba incluir.

Además, se deben observar los siguientes aspectos:

- Toda la **planificación**, así como todos los documentos correspondientes, **actas de las reuniones, descripción de los casos de uso, documentos de análisis, ficheros Visual Paradigm, ficheros de imputación de horas, ...)** **deben ser enlazados desde la wiki**, de modo que la wiki del proyecto sea la guía de lectura de la documentación, facilitando así la corrección¹.
- Es **obligatorio que se documenten todas las reuniones que los grupos de trabajo realicen como parte del trabajo de colaboración**. Todas las decisiones / compromisos que se obtengan se mapearán a diferentes *issues* en Github, y se monitorizará adecuadamente el avance mediante el control de un tablero Kanban simplificado en un *proyecto que se creará adhoc para la práctica*. Esto servirá para marcar la trazabilidad del proyecto, y **debe haber una coherencia entre la planificación realizada, los compromisos alcanzados en la reunión, los issues creados, y la secuencia de avances del proyecto** (marcados por la revisión de los compromisos en los proyectos). Es decir, los pasos que deberían darse son los siguientes a modo de ejemplo:

¹ A modo de ejemplo, os pondremos más adelante en este mismo documento algunos ejemplos de años anteriores.

- En la planificación realizada en la fase de Inicio se marca un hito específico (p.ej. se creará la clase `Persona.java v1.3.0` en la fecha 28/10²)
- En la reunión de trabajo celebrada en el 15/10 se toma y acepta la decisión que *Pepito López desarrollará la clase `Persona.java v1.3.0`*
- El administrador del proyecto crea la tarea en el proyecto “Desarrollo Módulo X” en el que está la tarea *Creación de la clase `Persona.java v1.3.0`* y creará un *issue “Creación de la clase `Persona.java v1.3.0`”* y se asignará a *Pepito López para que haga el commit correspondiente en torno al 28/10.*
- Pepito López trabajará a partir del día 15/10 (tras la reunión de coordinación) y hará el *commit* correspondiente en torno al 28/10 y si se diera la circunstancia el *pull request*.
- En la sección de *Insights* de Github se podrán consultar el avance del proyecto
- Debéis incorporar la cuenta de usuario de los profesores (profesor.aigomez@uclm.es para el grupo BC e Ismael.Caballero@uclm.es (IsmaelCaballero) para los grupos B y C) como **contributors del proyecto** para que puedan monitorizar el avance del estado de las prácticas.
- Para guiar mejor el desarrollo, es altamente recomendable que se utilicen las funcionalidades de gestión de proyecto disponible en Visual Paradigm (véase Figura 3). Se recuerda que **en cada iteración se trabajará exclusivamente en el o los casos de usos especificados**³. Además, y para ahorrar costes, se utilizará la funcionalidad de generación automática de código proporcionada por Visual Paradigm.
- Con respecto a la implementación, los alumnos deben implementar los módulos necesarios como para satisfacer todos los requisitos funcionales especificados, teniendo en cuenta la planificación realizada y el método de trabajo (iterativo, incremental, colaborativo).
- Teniendo en cuenta la aproximación 1:1 de la que partimos, se considerará que cada iteración va a generar inicialmente un componente, cuyo desarrollo debe establecerse en la planificación marcando adecuadamente las versiones que se espera ir consiguiendo – las versiones se nombrarán usando **Semantic Versioning**-. Al final en la última iteración se procederá a la integración de todos los componentes. **Cada uno de esos componentes se tratarán como si fuera un módulo de un proyecto Maven**. Por tanto, se deberá gestionar el desarrollo **como si fuera un proyecto multimódulo** (véase como ejemplo el capítulo 6 del libro “*Maven by examples*” (<http://books.sonatype.com/mvnex-book/reference/index.html>)). Esta gestión puede hacerse desde la línea de comandos, o bien usando los plugins adecuados del IDE que utilicéis (recomendablemente Eclipse, con el plugin m2Eclipse).
- El código resultante estará sujeto a la realización de las pruebas y a mantenimiento (objeto de las prácticas correspondientes a los temas 4 y 5)
- En la wiki se tiene que incluir una sección de **“Autoevaluación y Experiencia”** en donde **todos los miembros del grupo** incorporen una autoevaluación y una autocrítica⁴ de su experiencia en el proyecto.

² Las planificaciones deberían cumplirse en la medida de lo posible; no obstante, y como se ha dicho en clases de prácticas, lo normal es que esas fechas pueda variar por diversas razones: falta de experiencia, complicaciones tecnológicas, ... todas estas circunstancias son asumidas en la evaluación, pero los grupos de alumnos deben hacer un análisis sincero de las razones por las que no se ha cumplido la planificación, y proporcionar medios para paliar las desviaciones de la agenda, explicando cómo se va a resolver el problema para poder acabar a tiempo el proyecto.

³ Es decir, que **no vale con ponerse a hacer todos los casos de uso, o hacer todos los diagramas de clases a la vez, sino cuando le corresponda según la planificación realizada.**

⁴ Autocrítica no es necesariamente algo negativo: pueden ser cosas del tipo “*aprendí mucho*” o “*considero que las prácticas no me han valido para mucho*”, o “*creo que debería haber trabajado más*”, o “*ahora que hemos terminado la práctica, empiezo a entender muchas más cosas*”

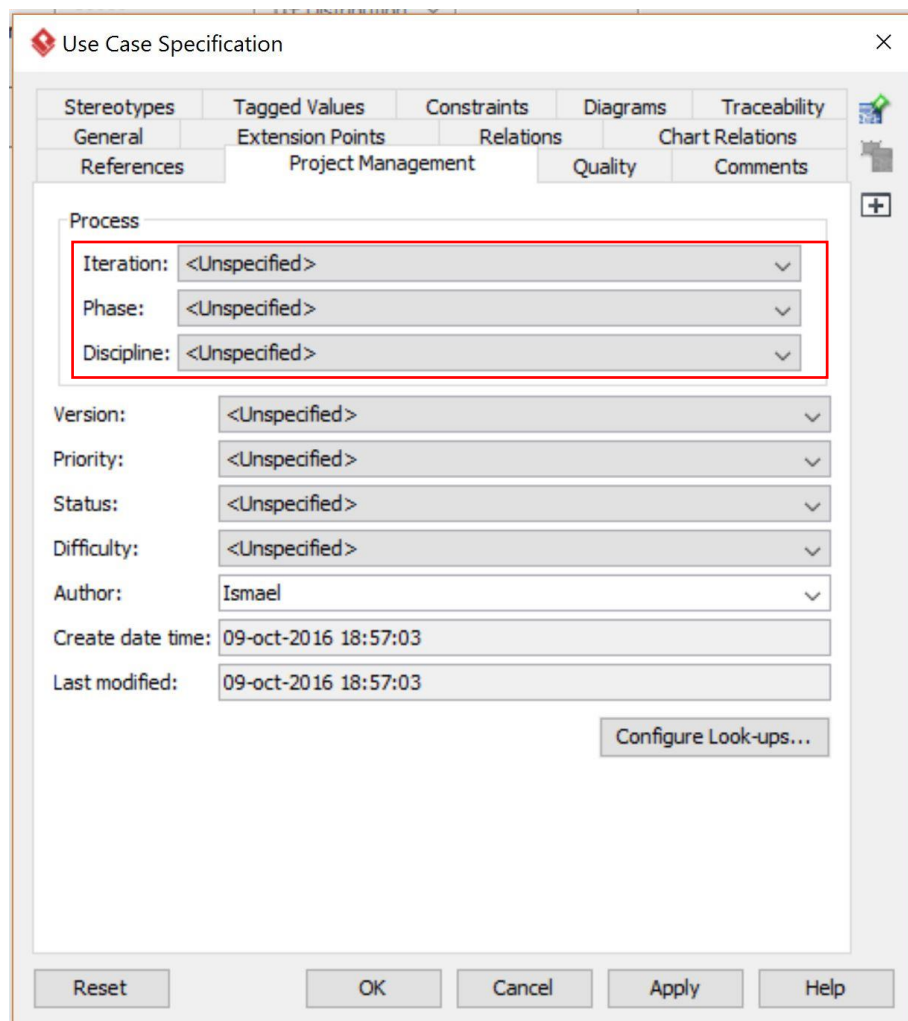


Figura 3: mecanismos de gestión de proyectos en Visual Paradigm.

Algunos aspectos importantes

La experiencia de la realización de prácticas de otros años nos ha permitido aprender una serie de lecciones que os transmitimos:

- **Aplicación del Proceso Unificado de Desarrollo:** Sería muy interesante que releverais este enunciado una vez que haya pasado un tiempo. Uno de los principales problemas con el que nos encontramos es que los alumnos no entienden bien la asignación de casos de usos a iteraciones. Por simplicidad os recomendamos que hagáis **1RF:1CDU y 1CDU:1Iteracion**. Pero esto es una recomendación -que reduce la aplicación el PUD a un ciclo en cascada - hasta que entendáis realmente que el desarrollo de un software debe agruparse en varios bloques (a los que llamamos módulos) que deben cumplir la regla de máxima coherencia y mínimo acoplamiento (Principios SOLID).
- **Programación Orientada a Interfaces, no a clases:** Como consecuencia de lo anterior, uno de los principales problemas con el que nos encontramos año a año es que los alumnos no disponen de conocimientos suficientes de programación como para entender la diferencia entre programación orientada a interfaces y programación orientada a clases. La naturaleza iterativa e incremental del PUD, y el hecho de estar dirigido por casos de uso, lleva necesariamente a un planteamiento en el que **el software debe ser agrupado en varios bloques** (a los que llamaremos módulos), que deben ser desarrollado – de forma independiente deseablemente- en cada una de las iteraciones de la

aplicación del PUD según se haya determinado en la planificación. Esta independencia obliga a un mínimo acoplamiento entre módulo que se resuelven mediante la declaración y uso de Interfaces y la aplicación del modelo multicapa (Patrón MVC – Modelo-Vista-Controlador). Cada uno de los módulos requiere paquetizar y usar adecuadamente las clases para que la comunicación se produzca a través de las interfaces correspondientes. Así que os animamos a repasar la creación de paquetes y a usarlos en el desarrollo. A fin de hacer una programación más profesional os animaríamos a usar algún framework de desarrollo tipo Spring que os simplificará mucho el trabajo una vez que hayáis superado la curva de aprendizaje mínima

- **Uso de Ramas en Github y de Maven.** Estos son de las características que más problemas suelen causar a la hora de realizar las prácticas. El problema nuevamente radica en que los alumnos abordan el desarrollo del software como un todo, sin tener en cuenta la arquitectura del sistema. Sin embargo, si los alumnos consiguieran entender que el desarrollo usando metodologías iterativas e incrementales (independientemente de si son estructuradas o ágiles) implica la descomposición de la aplicación en lo que venimos llamando módulo. Es muy importante que los módulos no dupliquen el código, sino que lo usen a medida que se necesita.
- **Abordar proyectos pequeños para entender bien los conceptos de Git y Maven.** La principal dificultad de la práctica está en asumir el cambio de paradigma (Iterativo e Incremental y Colaborativo). Si a esto se le añade la complejidad de herramientas como Git y Maven, puede llegar a ocurrir que los alumnos os sintáis confundidos y sobrepasados. Por esa razón, os animamos a que abordéis el aprendizaje de Git, Maven y otras herramientas fuera del contexto del proyecto, aplicándolo a proyectos pequeños. Una vez aprendidos y madurados estos conceptos, ya podréis aplicarlos al proyecto.

Tema 2. Gestión de Configuración

El objetivo de esta parte es la incorporación del Plan de Gestión de Configuración. La implementación de dicho plan debería quedarse reflejada en varios aspectos: quién aprueba los requisitos, quién se responsabiliza de cada componente, quién aprueba los componentes, el plan de versiones de los componentes y versiones de las construcciones.

Especialmente importante es el plan de versiones de cada componente que, debería realizarse usando la notación Semantic Versioning (<http://semver.org/>), y quedar reflejado de acuerdo con las recomendaciones indicadas en Git Flow, con la creación de las diferentes ramas.

Tema 3. Gestión de Calidad

El objetivo de este bloque de prácticas es la derivación de diversos requisitos tanto funcionales como no funcionales que podrían aparecer al considerar los aspectos de calidad de productos software (quality by design).

Bloque 2

Tema 4. Gestión de pruebas

El objetivo para las prácticas de este tema es desarrollar y ejecutar el plan de pruebas para el proyecto. En este sentido, se deben tener en cuenta los siguientes aspectos:

- El plan de prueba debe incluir casos de pruebas para alcanzar al menos un nivel de cobertura del 75%

- Se generarán mediante la integración de los plugins para Surefire y Jacoco de Maven el **informe de pruebas correspondientes**, que deberá aparecer enlazados en la wiki del proyecto. Se aportará más información cuando se alcance el tema de gestión de pruebas.

Tema 5. Gestión de mantenimiento

Tomando como punto de partida el informe de pruebas obtenido en el informe anterior se **trata de corregir los errores en el código y o de arreglar algunos aspectos de limpieza del código (*clean code*)** Para ello se realizará un plan de mantenimiento enfocado al mantenimiento del sistema, que partirá de la **creación de un informe automatizado generado con Maven mediante la inclusión de los diferentes plugins: PMD/Findbugs**. Se aportará más información cuando comiencen las prácticas del Tema 5.

Herramientas recomendadas necesarias.

- 1) Entorno **Eclipse con plugins de Maven y de Git**.
- 2) Servidor de base de datos **MySQL** (o máquinas virtuales con los SGBDRs desplegados y los puertos correspondiente del firewall abiertos).
- 3) Cliente de base de datos **MySQL Workbench**
- 4) **Visual Paradigm**

Ejemplo de proyectos de años anteriores:

En los siguientes enlaces a GitHub podéis encontrar ejemplos de años anteriores. Mirad la estructura del wiki, los ficheros creados, ... Os pueden aportar información valiosa, pero recordad que las soluciones que aportaron los compañeros no tienen por qué coincidir con la forma en la que lo haríais vosotros (cread vuestro propio estilo), aunque son una buena base de partida. También permitidme recordaros que en estos años no explotamos mucho la opción de gestionar los proyectos, por lo que no están reflejados en estos ejemplos.

- <https://github.com/guillecchm/SoftMobile.git>
- <https://github.com/helendiaz1998/EspartaSoft.git>
- <https://github.com/Gerole10/JuciyCode.git>
- <https://github.com/manuvillalba-uclm/ISO-II-Alumavic.git>
- <https://github.com/noeliagranados98/ISOII.git>

Hay abierta una wiki en la sección del proyecto para que podáis enlazar el enlace al repositorio⁵ github de vuestro proyecto.

⁵ Recordad que Github no permite borrar nada, por lo que todo lo que subáis, se quedará en vuestro repositorio para siempre... en su lugar y para hacer pruebecillas, os recomiendo que uséis otros repositorios auxiliares.