**Documento de gestión de proyecto**

Autor: Víctor García-Bermejo Mazorra

Fecha: 8/6/2020

Índice

[1. Introducción 3](#_Toc74141473)

[1.1. Objeto 3](#_Toc74141474)

[1.2. Ámbito de la aplicación 4](#_Toc74141475)

[2. Aspectos generales del proyecto 4](#_Toc74141476)

[2.1. Objetivos del sistema 4](#_Toc74141477)

[2.2. Descomposición en subsistemas 5](#_Toc74141478)

[3. Catálogo de requisitos 7](#_Toc74141479)

[3.1. Requisitos funcionales 7](#_Toc74141480)

[3.1.1. Subsistema de visualización y edición de diagramas BPMN 7](#_Toc74141481)

[3.1.2. Subsistema de visualización y edición de diagramas UML 7](#_Toc74141482)

[3.1.3. Subsistema de visualización y edición de diagramas de redes de Petri 8](#_Toc74141483)

[3.1.4. Subsistema de transformación de BPMN a red de Petri 8](#_Toc74141484)

[3.1.5. Subsistema de transformación de UML a red de Petri 9](#_Toc74141485)

[3.1.6. Subsistema de análisis 9](#_Toc74141486)

[3.2. Requisitos no funcionales 10](#_Toc74141487)

[3.2.1. Requisitos de interfaz y usabilidad 10](#_Toc74141488)

[3.2.2. Requisitos operacionales 10](#_Toc74141489)

[3.2.3. Requisitos de documentación 10](#_Toc74141490)

[3.2.4. Requisitos de mantenibilidad y portabilidad 10](#_Toc74141491)

[3.2.5. Requisitos de verificación y fiabilidad 10](#_Toc74141492)

[4. Gestión del Proyecto 11](#_Toc74141493)

[4.1. Estimaciones del Sistema de Software 11](#_Toc74141494)

[4.1.1. Puntos de Función sin Ajustar 11](#_Toc74141495)

[4.1.2. Puntos de Función Ajustados 13](#_Toc74141496)

[4.2. Ciclo iterativo incremental 14](#_Toc74141497)

[5. Seguimiento y control 17](#_Toc74141498)

[5.1. Seguimiento del avance 17](#_Toc74141499)

[6. Conclusiones 18](#_Toc74141500)

[6.1. Actualidad del proyecto 18](#_Toc74141501)

# Introducción

## Objeto

Este documento tiene como objetivo describir los requisitos funcionales y no funcionales que el proyecto de este Trabajo de Fin de Máster tendrá que cumplir para su realización y, en base a estos, proponer una gestión para el desarrollo de los mismos. Como cualquier producto software, en base a los requisitos que se propondrán, se desarrollará una aplicación dividida en subsistemas que incluirán toda funcionalidad propuesta.

El producto software que se desarrollará consistirá en un plugin cuyas principales funcionalidades se centran en la creación de sistemas utilizando diagramas Unified Model Language (UML) o Business Process Model and Notation (BPMN); en la transformación de estos a redes de Petri estocásticas temporizadas, y finalmente en su análisis para conocer sus propiedades dinámicas.

Para el desarrollo de este proyecto, se seguirá un modelo de ciclo de vida incremental e iterativo en el que se distinguirán tres incrementos base, desarrollando cada uno de los subsistemas propuestos. Estos incrementos contarán con las fases de prerrequisitos, análisis de requisitos, diseño, implementación y pruebas. Cabe destacar que, como todo el proyecto será hecho por una sola persona, los incrementos no estarán solapados y cada uno de ellos empezará una vez termine el anterior.

Teniendo en cuenta esto, en la primera sección de este documento se propone una descripción de alto nivel relacionada con el ámbito y alcance que ocupará este Trabajo de Fin de Máster. Se propondrán una serie de funcionalidades que debe tener la aplicación, y una variedad de elementos que no la compondrán con el objetivo de marcar los límites de la aplicación.

En la segunda sección define el proyecto y da una descripción profunda de cada uno de los sistemas que lo componen y sus funcionalidades.

En el tercer apartado se expone un catálogo que acoge todos los requisitos funcionales y no funcionales que conformarán este proyecto.

Durante el cuarto apartado se preparará un plan de gestión del proyecto basado en el ciclo de vida incremental y utilizando los requisitos propuestos anteriormente para el cálculo de los puntos de función sin ajustar y ajustados.

En el apartado cinco se propondrán una serie de reuniones ligadas a la finalización de los ciclos de desarrollo de software propuestos en el plan de gestión del proyecto con la idea de generar un seguimiento y un control al avance del proyecto.

Finalmente, todo el documento será acompañado por conclusiones donde se hará un breve recorrido sobre todo el documento comentando resultados y se añadirá una opinión de la situación del proyecto.

## Ámbito de la aplicación

El proyecto será un plugin que será compatible con el entorno de programación Eclipse (versión de marzo de 2021) con las siguientes funcionalidades:

* Generación de diagramas BPMN.
* Generación de diagramas UML.
* Generación de diagramas de redes de Petri.
* Transformación de diagrama BPMN a red de Petri.
* Transformación de diagramas UML a red de Petri.
* Análisis de redes de Petri.
* Almacenamiento de los análisis realizados.
* Análisis de diagramas BPMN en base a análisis de redes de Petri.
* Análisis de diagramas UML en base a análisis de redes de Petri.

El proyecto no incluirá las funcionalidades:

* Generación de diversos tipos de diagramas en el mismo esquema.
* Transformación de red de Petri a diagramas UML.
* Transformación de red de Petri a diagrama BPMN.
* Análisis de forma directa sobre diagramas UML.
* Análisis de forma directa sobre diagramas BPMN.

# Aspectos generales del proyecto

En esta sección se analizarán los objetivos principales del proyecto, para lo cual se describen las necesidades que han dado lugar a la realización de este. También se analizarán los subsistemas en que se divide la aplicación mediante una descripción detallada.

## 2.1. Objetivos del sistema

Este Trabajo de Fin de Máster propone la generación de una serie de módulos y *plugins* para el análisis de variabilidad de UML y BMPN basado en Líneas de Producto de Redes de Petri. ya que, hoy en día, no existe una herramienta que, en base al diseño de diagramas de líneas de productos BPMN o diagramas de líneas de productos UML se haga un análisis extenso de su seguridad y variabilidad de estos.

Teniendo en cuenta esto, uno de los *plugins* que compondrá el proyecto, deberá ser una herramienta de diseño que permita la visualización, creación y la edición de diagramas basados en BPMN, UML. Adicionalmente, también se debería permitir la adicción a cualquiera de los diseños de los diagramas de un modelo de variabilidad que sirva para generar líneas de productos en base a los diagramas de tal forma que, cada elemento pueda corresponderse a una *feature* de un *feature model.*

## 2.2. Descomposición en subsistemas

El sistema estará compuesto por una serie de subsistemas independientes de otros y cuya función principal será el diseño de diversos diagramas; y por otros de carácter dependiente y que generan relación entre los subsistemas independiente mediante transformaciones o análisis. A continuación, se detallan sus principales características.

El subsistema de visualización y edición de diagramas BPMN se encarga de:

* Diseño de diagramas de BPMN mediante una herramienta de edición. Desde esta herramienta se crearán los diversos elementos que conforman este tipo de diagramas. La creación del archivo del diagrama no estará contemplada en dicha herramienta.
* Validación de los diagramas BPMN generados. Se establecerán unas normas básicas acerca de los elementos y sus relaciones que serán comprobados por el subsistema con la idea de verificar si el sistema sintácticamente es correcto.

El subsistema de visualización y edición de diagramas UML se encarga de:

* Diseño de diagramas de UML mediante una herramienta de edición. Cuando se diseñe un diagrama se deberá elegir entre las diversas posibilidades que ofrece el metamodelo de UML: clases, componentes, estados, secuencia, casos de uso y despliegue. El motivo de esto es debido a que la herramienta se deberá adaptar al diagrama que se pretende diseñar proponiendo la creación de los elementos del tipo concreto. La creación del archivo del diagrama no estará contemplada en dicha herramienta.
* Validación de los diagramas UML generados. Se establecerán unas normas básicas acerca de los elementos y sus relaciones de cada uno de los tipos de diagramas que serán comprobados por el subsistema con la idea de verificar si el sistema sintácticamente es correcto.

El subsistema de visualización y edición de diagramas de redes de Petri se encarga de:

* Diseño de redes de Petri. Existirá una herramienta de creación y edición de los diversos elementos de un diagrama de este tipo.
* Validación sintáctica de las redes de Petri. Existirán una serie de reglas basadas en las relaciones de los elementos de las redes de Petri.

Todos estos subsistemas anteriores también añadirán en su funcionamiento la posibilidad de generar líneas de productos de los diagramas mediante sistemas de variabilidad y condiciones de presencia. Por ello, todas las herramientas también tendrán un menú donde se permitirá la adición de características y de relaciones que asociarán estas características a los elementos de cada diagrama.

Tras finalizar los subsistemas relacionados con el diseño y la visualización grafica de los elementos de los distintos diagramas acogidos en la asignatura, se describen los subsistemas de transformación y el subsistema de análisis ya que estos dependen de la existencia de los descritos anteriormente.

El subsistema de transformación BPMN se encarga de:

* Transformaciones de diagramas de BPMN a redes de Petri. Los distintos elementos o conjuntos de elementos de un diagrama basado en BPMN serán relacionados de la forma más directa posible con elementos de una red de Petri, haciendo una transformación model-to-model.
* Transformación de líneas de Productos de BPMN en líneas de Productos de redes de Petri. Es de vital importancia mantener la variabilidad y/o adaptarla tal que se hagan representaciones correctas de líneas de productos en los diagramas de redes de Petri, puesto que las posibilidades que se pueden comprender dentro de una línea de Productos varían de forma drástica cualquier diagrama.

El subsistema de transformación UML se encarga de:

* Transformaciones de diagramas de UML a redes de Petri. En este caso, la transformación no es trivial y para ello, todos los diagramas que se generen acerca de un sistema serán transformados a una única red de Petri.
* Transformación de líneas de Productos de UML en líneas de Productos de redes de Petri.

El subsistema de análisis se encarga de:

* Teniendo como entrada redes de Petri o líneas de Producto de redes de Petri analizar la seguridad de una red de Petri.
* Encontrar posibles incoherencias en las redes de Petri resultantes para poder efectuar de forma precisa el análisis.
* Mostrar los resultados a los usuarios con el objetivo de que puedan mejorar sus sistemas.

# Catálogo de requisitos

## 3.1. Requisitos funcionales

### 3.1.1. Subsistema de visualización y edición de diagramas BPMN con variabilidad (SP)

1. El usuario podrá crear y editar diagramas conformes con BMPN con variabilidad.
   1. Los diagramas creados se almacenarán
      1. en formato .bpmn2
      2. Como se guarda la variablidad…
   2. La herramienta permitirá la creación y adición de objetos de flujo, objetos de conexión, carriles de nado y artefactos.
      1. Dentro de los objetos de flujo encontraremos eventos, actividades y compuertas.
      2. Dentro de los objetos de conexión encontraremos uniones denominadas flujos de secuencia, de mensaje y asociaciones.
      3. Dentro de los carriles de nado se podrán crear piscinas y carriles.
      4. Dentro de los artefactos se encuentran objetos de datos, grupos y anotaciones.
2. En la visualización gráfica de los diagramas BMPN existirá una herramienta de adición elementos.
3. Durante todo el proceso de diseño, habrá en la herramienta de creación elementos relacionados con la variabilidad para poder definir líneas de productos BPMN basadas en el diagrama que se está diseñando.
   1. Los elementos de variabilidad se corresponden a condiciones de presencia, expresiones y las relaciones con los elementos del diagrama BPMN.
   2. Los ficheros donde se guardarán estos elementos serán de extensión .vrb.

### 3.1.2. Subsistema de visualización y edición de diagramas UML (SU)

1. El usuario podrá crear y editar diagramas conformes con UML con variabilidad.
   1. Los diagramas creados se almacenarán en formato .uml
2. En la visualización de los diagramas UML existirá una herramienta de adición elementos.
   1. La herramienta para diagramas UML permitirá crear diagramas de clases, de componentes, de estados, de secuencia, de casos de uso y de despliegue.
      1. Dentro de la creación de diagramas de clase, se podrán generar clases, enumeraciones, asociaciones, composiciones, agregaciones, atributos y métodos. Todos los elementos tendrán unas propiedades para su edición.
      2. Dentro de los diagramas de componentes se podrán crear componentes, interfaces y relaciones de dependencia.
      3. Dentro de los diagramas de estados se podrán crear estados compuestos, pseudoestados de opción, eventos, puntos de salida, primer estado, estados, subestados, estado final, transiciones entre estados y disparadores.
      4. Dentro de los diagramas de secuencia se podrán crear líneas de vida, mensajes actores y relaciones que determinarán el orden temporal en el que transcurre toda la secuencia.
      5. Dentro de los diagramas de casos de uso se podrán crear sistemas, casos de uso, actores y las relaciones de inclusión, exclusión y líneas de comunicación.
      6. Dentro de los diagramas de despliegue se podrán crear nodos, componentes y asociaciones.
3. Durante todo el proceso de diseño, habrá en la herramienta de creación elementos relacionados con la variabilidad para poder definir líneas de productos UML basadas en el diagrama que se está diseñando.
   1. Los elementos de variabilidad se corresponden a condiciones de presencia, expresiones y las relaciones con los elementos del diagrama UML.
   2. Los ficheros donde se guardarán estos elementos serán de extensión .rvb.

### 3.1.3. Subsistema de visualización y edición de diagramas de redes de Petri

1. El usuario podrá crear y editar diagramas que representen redes de Petri.
   1. Los diagramas creados se almacenarán en formato .petrinets
2. En la visualización de los diagramas de redes de Petri existirá una herramienta de adición elementos.
   1. La herramienta para redes de Petri permitirá crear lugares transiciones y arcos.
3. Durante todo el proceso de diseño, habrá en la herramienta de creación elementos relacionados con la variabilidad para poder definir líneas de productos de redes de Petri basadas en el diagrama que se está diseñando.
   1. Los elementos de variabilidad se corresponden a condiciones de presencia, expresiones y las relaciones con los elementos de una red de Petri.
   2. Los ficheros donde se guardarán estos elementos serán de extensión .vrb.

### 3.1.4. Subsistema de transformación de BPMN a red de Petri

1. El sistema permitirá obtener utilizando como entrada un diagrama BPMN una red de Petri equivalente.
   1. La transformación de BPMN consistirá en la transformación directa de un diagrama BPMN a una red de Petri.
   2. El fichero de entrada que contendrá el diagrama BPMN tendrá el.bpmn2, una extensión basada en XMI.
   3. El fichero de salida que contendrá la red de Petri resultante tendrá la extensión .petrinets.
2. El sistema adaptará la variabilidad con el objetivo de que todos los posibles productos resultantes de una línea de productos de BPMN sean semejantes a los resultantes de la transformación a una línea de productos de redes de Petri.
3. La transformación aparecerá como un menú contextual con un solo archivo de extensión .bpmn2 seleccionado.
   1. No se aceptará la selección de varios archivos a la vez.
   2. En caso de ser línea de Productos BPMN se buscará el archivo de extensión .vrb cuyo nombre sea el mismo que el archivo que se va a transformar y se obtendrá como salida otro con el mismo nombre que la red de Petri resultante.

### 3.1.5. Subsistema de transformación de UML a red de Petri

1. El sistema permitirá obtener utilizando como entrada varios diagramas UML una red de Petri equivalente.
   1. La transformación de UML consistirá en una transformación utilizando varios diagramas a la vez para obtener como resultado una red de Petri.
      1. Diagrama de clases.
      2. Diagrama de componentes.
      3. Diagrama de estados.
      4. Diagrama de secuencia.
      5. Diagrama de casos de uso.
      6. Diagrama de despliegue.
   2. Los ficheros de entrada que contendrán los diagramas UML tendrán la extension .uml.
   3. El fichero de salida que contendrá la red de Petri resultante tendrá la extensión .petrinets.
2. El sistema adaptará la variabilidad con el objetivo de que todos los posibles productos resultantes de una línea de productos de UML sean semejantes a los resultantes de la transformación a una línea de productos de redes de Petri.
   1. Se tratarán todas las líneas de producto de cada archivo por separado con la idea de finalmente unificarlas en la línea de productos de la red de Petri resultante.
3. La transformación aparecerá como un menú contextual cuando se seleccionan uno o varios ficheros de con extension .uml.
   1. El programa marcará error en todos aquellos casos que no cumplan el número adecuado de diagramas de entrada para su ejecución.
      1. Se exigirá en el mensaje de error que se escoja un diagrama de cada tipo.
   2. En caso de ser línea de Productos UML se buscarán los archivos de extensión .vrb cuyo nombre sea el mismo que los archivos que se van a transformar y se obtendrá como salida otro con el mismo nombre que la red de Petri resultante en el que se unirán todas las líneas de los diversos tipos de diagramas UML.

### 3.1.6. Subsistema de análisis

1. Cualquier usuario podrá acceder al análisis de las redes de Petri debido a que, cualquier usuario que haya creado un diagrama debe poder saber si el análisis de la misma da un resultado razonable.
2. Este subsistema tendrá como entrada archivos que contendrán redes de Petri.
3. El resultado de todo el análisis hecho sobre la red de Petri que trate, será guardado en un archivo en formato .txt.
4. Este subsistema dentro de la plataforma de eclipse, será creado como un *plugin* donde se preguntará al usuario cual es el fichero que se desea analizar, mostrando una lista de ficheros en el espacio de trabajo de la extensión .petrinets.

## 3.2. Requisitos no funcionales

### 3.2.1. Requisitos de interfaz y usabilidad

1. La interfaz gráfico de los subsistemas de visualización y diseño de diagramas se intentará simplificar al máximo puesto que tiene una gran cantidad de botones y elementos en la herramienta de diseño, intentando ser distinguibles entre sí mediante agrupaciones de tipo.
2. La carga visual de los menús de transformación y análisis será lo más simple posible teniendo en cuenta los elementos que han de mostrar.

### 3.2.2. Requisitos operacionales

1. En los subsistemas dedicados a la visualización y al diseño los elementos serán guardados cuando es usuario de esa orden utilizando atajos de teclado o mediante la interfaz de eclipse.
   1. Al cerrar Eclipse sin guardar los documentos con cambios, el programa preguntara si guardar los cambios.
2. Cualquier error en la transformación y en el análisis se indicará mediante un dialogo emergente y quedará registrado en el log de Eclipse.

### 3.2.3. Requisitos de documentación

1. Durante la fase de diseño de este proyecto, se generarán diagramas de casos de uso, de clases y de secuencia relacionados con los subsistemas y sus interacciones entre ellos.
2. Al finalizar el proyecto, se generará un manual de instrucciones para que cualquier usuario sea capaz de utilizarlo.

### 3.2.4. Requisitos de mantenibilidad y portabilidad

1. La aplicación tendrá soporte en el entorno de programación de Eclipse, en la versión de marzo de 2021. Se entiende que los plugin será compatible con versiones posteriores siempre.

### 3.2.5. Requisitos de verificación y fiabilidad

1. La aplicación gestionará los errores y las situaciones anómalas mediante la eliminación de los ficheros que queden sin ser finalizados (rollback).

# 4. Gestión del Proyecto

En esta sección analiza las propiedades principales del plan de proyecto creado para el desarrollo de los subsistemas y los requisitos mencionados anteriormente con la idea de crear una herramienta para Eclipse. A continuación, se hará una estimación de los puntos de función del sistema a desarrollar y se propone una organización del proyecto en base a la estimación. Además, se propondrá la asignación y el orden de las tareas a realizar y otros elementos relevantes como los costes.

## 4.1. Estimaciones del Sistema de Software

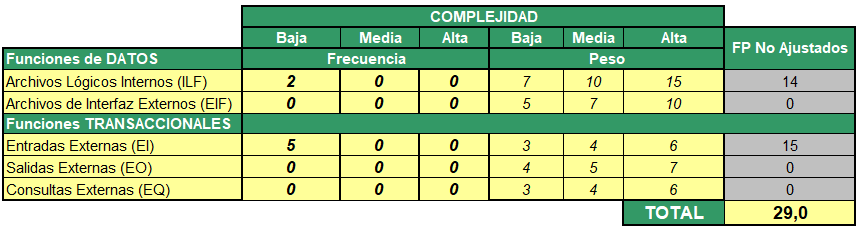
Las estimaciones de la carga del trabajo y complejidad del mismo resultan decisivas a la hora de realizar un plan de proyecto ajustado a los límites de tiempo reales.

Debido a esto, se ha estimado el tamaño del proyecto mediante la técnica de estimación por puntos de función.

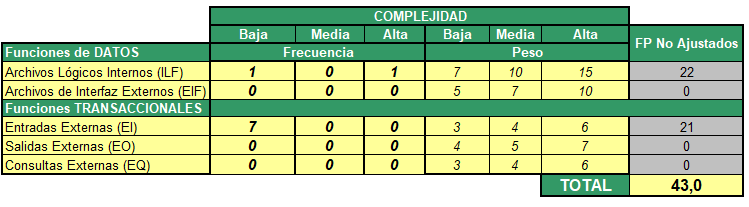
### 4.1.1. Puntos de Función sin Ajustar

Una vez se proponen los requisitos del sistema, mediante el análisis de puntos de función, se han obtenido los datos correspondientes a su complejidad estimada. Se adjuntan las tablas de los distintos subsistemas junto con las estimaciones en términos de punto de función.

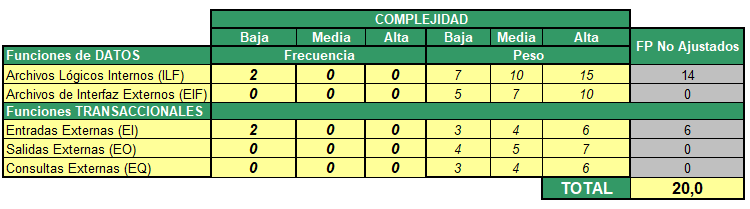
**Subsistema de visualización y edición de diagramas BPMN**



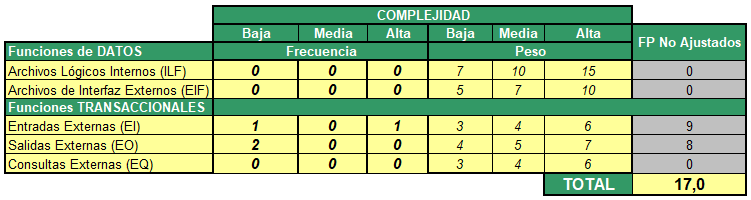
**Subsistema de visualización y edición de diagramas UML**



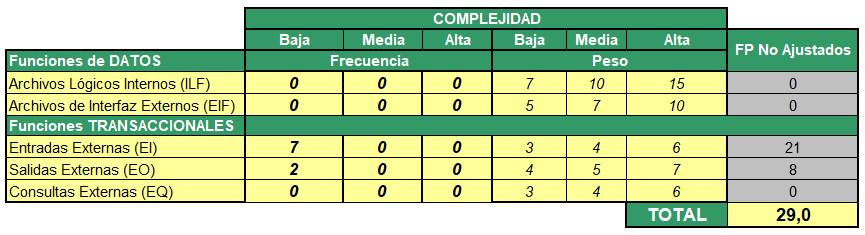
**Subsistema de visualización y edición de diagramas de redes de Petri**



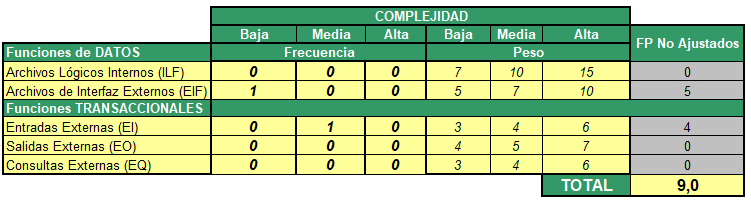
**Subsistema de transformación de BPMN a red de Petri**



**Subsistema de transformación de UML a red de Petri**



**Subsistema de análisis**



### 4.1.2. Puntos de Función Ajustados

Una vez obtenidos los puntos de función sin ajustar es necesario normalizar el resultado. Para ello se calcula un factor de ajuste de todo el proyecto mediante la siguiente tabla:

|  |  |
| --- | --- |
| Factores de Complejidad | CF |
| Comunicaciones de datos | 3 |
| Funciones distributivas | 2 |
| Prestaciones | 0 |
| Gran uso de la configuración | 1 |
| Velocidad de transacciones | 0 |
| Entrada de datos online | 1 |
| Diseño para la eficiencia del usuario | 3 |
| Actualización de datos online | 0 |
| Complejidad del proceso L.I.A. | 4 |
| Reutilización | 4 |
| Facilidad de instalación | 2 |
| Facilidad de operación | 3 |
| Múltiples localizaciones | 0 |
| Facilidad de cambio | 3 |
| Total | 26 |

Una vez obtenida la complejidad total del proyecto, es necesario ajustar los puntos de función con forme a los resultados anteriores. Para ello, mediante la siguiente formula obtendremos finalmente el factor de ajuste que utilizaremos más adelante:

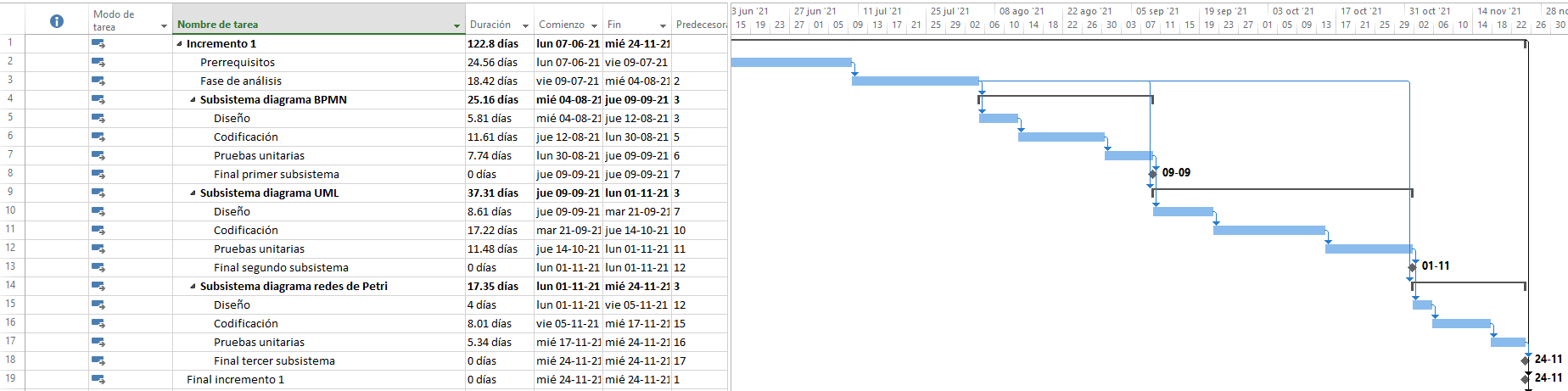
Conociendo el factor de ajuste, obtenemos los puntos de función ajustados reflejados en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Subsistemas | PFSA | PFA |
| Visualización y edición de diagramas BPMN | 29 | 26,39 |
| Visualización y edición de diagramas UML | 43 | 39,13 |
| Visualización y edición de diagramas de redes de Petri | 20 | 18,2 |
| Transformación de BPMN a redes de Petri | 17 | 15,47 |
| Transformación de UML a redes de Petri | 29 | 26,39 |
| Análisis | 9 | 8,19 |
| Total | 147 | 133,77 |

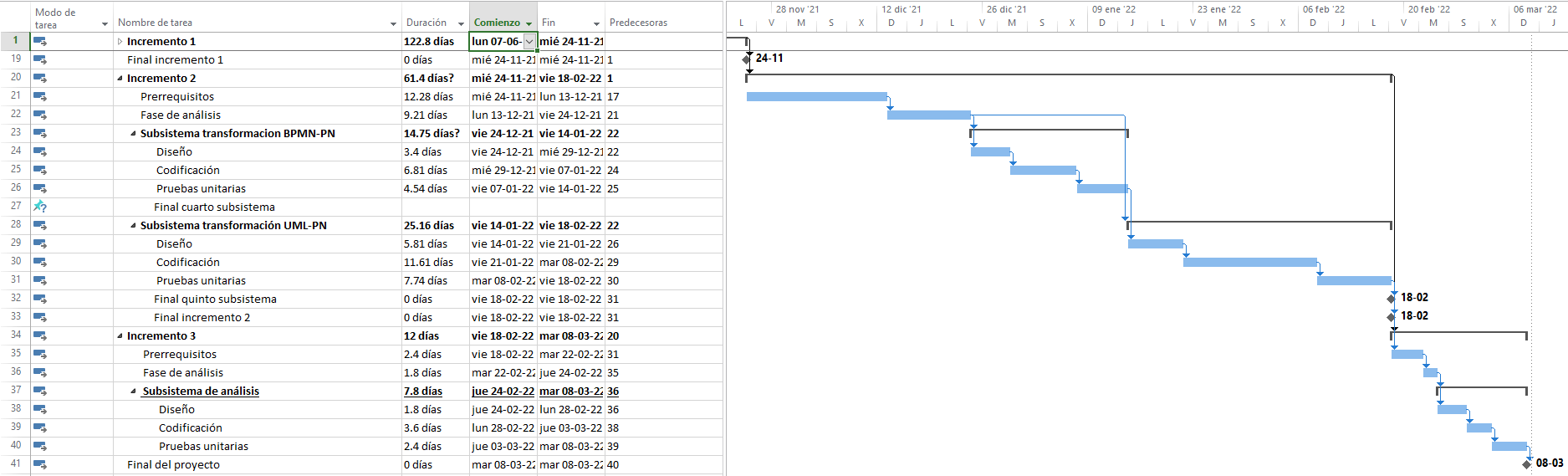
## 4.2. Ciclo iterativo incremental

Una vez obtenidos los puntos de función y sabiendo que el valor estándar de jornadas de trabajo por puntos de función es de 1,4667 llegamos a la conclusión de que el número de días laborables que se dedicaría al desarrollo de este proyecto es de 196,2. En este tiempo, se propone un ciclo incremental iterativo donde habrá seis iteraciones basadas en el número de subsistemas de la aplicación y cuyos tramos serán definidos por las fases del ciclo de vida del software.

Incremento 1:



Incrementos 2 y 3:



Como se puede apreciar en los diagramas, a lo largo de todo el desarrollo del proyecto, en ningún momento se superponen tareas y siempre se espera a finalizar las predecesoras. Esto se debe a que solo se dispone de un recurso y se ha decidido que se centre en una única tarea a la vez. Otro elemento que destacar del diagrama de Gantt es la división de incrementos. Para abordar el proyecto se propuso una división en tres incrementos donde se agruparían los sistemas por semejanza en sus características software:

* En el incremento 1 se propone desarrollar los subsistemas de visualización y edición de diagramas BPMN, visualización y edición de diagramas UML, y visualización y edición de diagramas de redes de Petri. En este caso y como el nombre de los subsistemas indican, se agruparon debido a que en todos los casos se pretendió desarrollar herramientas de diseño para los distintos tipos de diagramas. Este incremento durará 122,8 jornadas laborables.
* En el incremento 2 se propone desarrollar los subsistemas de transformación de BPMN a redes de Petri y de UML a redes de Petri. Éste durara 64,1 días.
* En el incremento 3 y por tanto para finalizar el proyecto, se propone desarrollar el subsistema de análisis de redes de Petri. Este incremento durará 12 días laborables.

Todos estos incrementos seguirán la misma metodología de trabajo. Ésta consistirá en una fase inicial donde se trabajarán una serie de prerrequisitos como la búsqueda de información y la adquisición de conocimiento acerca de los elementos que compondrán cada sistema de una forma más concreta que la propuesta en este documento. Tras esta fase, se ejecutará una fase común de análisis donde se revisarán los requisitos funcionales y no funcionales de cada subsistema que compone el incremento. Llegado este punto, se desarrollará, codificará y probará cada subsistema de forma individual.

Finalmente añadimos que, a pesar de un desarrollo de 196 jornadas laborables, puesto que el proyecto es un Trabajo de Fin de Máster hecho por Víctor García-Bermejo Mazorra, el coste será de un total de 0 euros.

# 5. Seguimiento y control

Puesto que el método de desarrollo propuesto para la resolución del proyecto debe haber un seguimiento estricto acerca del cumplimiento de los incrementos y las fases del desarrollo del producto software. Por ello, en este apartado se propone cómo seguir el avance del proyecto y una validación de cada subsistema.

## 5.1. Seguimiento del avance

Antes de la puesta en marcha del proyecto, se hará una reunión entre el único integrante de este Trabajo de Fin de Máster que será el principal desarrollador y el tutor que actuará como el cliente del producto software para llegar a un acuerdo acerca del desarrollo. Una vez aceptado el acuerdo, el proyecto dará comienzo.

Una vez comenzado el proyecto, se propondrá una serie de hitos basados en la finalización del ciclo de desarrollo software en cada subsistema, incremento o proyecto. Por ello, además de hacer reuniones semanalmente para el control del trabajo, se propondrán reuniones al final de la realización de hitos que se ven reflejados en esta tabla:

|  |  |
| --- | --- |
| Hito/s de la reunión | Fecha estimada |
| Finalización primer subsistema | 9/09/2021 |
| Finalización segundo subsistema | 1/11/2021 |
| Finalización tercer subsistema y primer incremento | 24/11/2021 |
| Finalización cuarto subsistema | 14/1/2022 |
| Finalización quinto subsistema y segundo incremento | 18/2/2022 |
| Finalización sexto subsistema, tercer incremento y proyecto | 8/3/2022 |

En estas reuniones se hará una validación por parte del cliente de cada subsistema con el objetivo de comprobar si todos los requisitos funcionales de carácter anterior a la reunión han sido satisfechos de forma adecuada mediante pruebas unitarias y otras comprobaciones acerca de requisitos no funcionales. En el caso de que no sea así, se marcarán como un objetivo para una de las reuniones semanales cercana a esa fecha.

# 6. Conclusiones

Este documento tiene como objetivo la gestión del proyecto del Trabajo de Fin de Máster del alumno Víctor García-Bermejo Mazorra. Para ello se hace un análisis del producto software que se va a generar como resultado en el que se pone un contexto y un ámbito al desarrollo representado finalmente en un catálogo de requisitos funcionales divididos en seis subsistemas y no funcionales del producto software.

Para paliar la necesidad de cumplir los requisitos propuestos, se hizo una estimación de cuanto se tardaría en hacer todo el proyecto dando como resultado 196,2 jornadas laborables. Este tiempo se dividió en tres incrementos en base a la semejanza que había entre los distintos subsistemas. En el primer incremento se desarrollarían los subsistemas orientados a diseño y visualización de diagramas y duraría 122,8 días. En el segundo incremento se trabajarían los subsistemas cuyo objetivo principal fuera la transformación de diagramas a redes de Petri y duraría 61,4 días. Finalmente, en el tercer incremento se implementaría un subsistema de análisis de redes de Petri y tendría una duración de 12 días.

Acompañando a todo este ciclo de desarrollo incremental iterativo, se propone un seguimiento basado en reuniones que tienen que están relacionadas con la resolución de hitos. Estos hitos marcaran la finalización de subsistemas, incrementos o el proyecto.

## 6.1. Actualidad del proyecto

A pesar de haber hecho el plan de proyecto a estas alturas, e indicando que la fecha de inicio es actual, el proyecto ha seguido un desarrollo constante donde se han ido cumpliendo prerrequisitos y análisis de varios de los subsistemas. Por otro lado, para la estimación de tiempo todos los subsistemas se han tratado de tal forma que se deban empezar desde cero. Sin embargo, ya hay componentes hechos en todos los casos por otros desarrolladores y se van a intentar utilizar con la idea de acortar periodos de tiempo.

Por estos motivos y por otros tantos, desde mi punto de vista pienso que la estimación del proyecto no es correcta para mi caso concreto y que debería plantearse de otra forma.