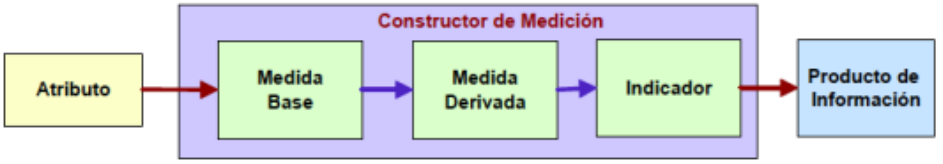
	Calidad, Medición y Estimación de Producto y Proceso Software	4º Grado en Ingeniería Informática Itinerario de Ingeniería del Software
	Examen Final de Teoría Convocatoria de febrero	10/2/2022

Ejercicio 1. PSM y SCRUM (2.5 puntos)



La empresa ACME desea elaborar un Plan de Medición de Software, siguiendo el estándar ISO/IEC 15939:2009. Conocedores de tu amplia experiencia en este ámbito adquirida en las prácticas de la asignatura CMEPPS, deciden contratarte para su puesta en marcha.

Tradicionalmente ACME ha usado el marco de trabajo SCRUM para la gestión y organización del trabajo, por lo que dispone de todos los datos de proyectos previos sobre estimación, historias de usuario, sprints y velocidades.

La empresa dispone de las siguientes mediciones por proyecto:

M1 Tamaño estimado previamente, midiéndose en puntos-historia globales de la pila del producto (*product backlog*)
M2 Iteraciones (*sprints*) realizados.
M3 Velocidad, promediada entre las iteraciones realizadas.
M4 Se caracterizan los proyectos en dos tipos de arquitectura, aplicaciones web basadas en frameworks PHP y proyectos basados en Enterprise Java.
M5 Complejidad ciclomática (media por clase) mediante cálculo de McCabe.
M6 Métodos ponderados por clase (media por clase).
M7 Defectos reportados por clientes después del despliegue.
M8 Esfuerzo medido en horas reales del equipo.
M9 Valor aportado, promediado entre las iteraciones realizadas.

Las necesidades de información que nos planteamos podrían formularse de la siguiente manera:

- Grado de avance del proyecto en el tiempo,

Usando PSM se desean plantear indicadores que, en base a los datos históricos, nos indiquen si el proyecto nuevo avanza adecuadamente, si disminuye el esfuerzo pendiente en el tiempo previsto.

Siguiendo el marco de trabajo que define el estándar PSM:

1) Explique todo el proceso que seguiría dentro del plan (fases y tareas a realizar) **(1 punto)**

Como se puede apreciar en la ilustración que aparece en el anexo, lo primero sería conseguir un **compromiso de la dirección** en el apoyo al plan en forma de recursos y a las disposiciones que de él salgan.

A continuación, una vez determinadas las necesidades de información, que en este caso son la *medición del grado de avance del proyecto en el tiempo*, realizamos la **planificación de la**

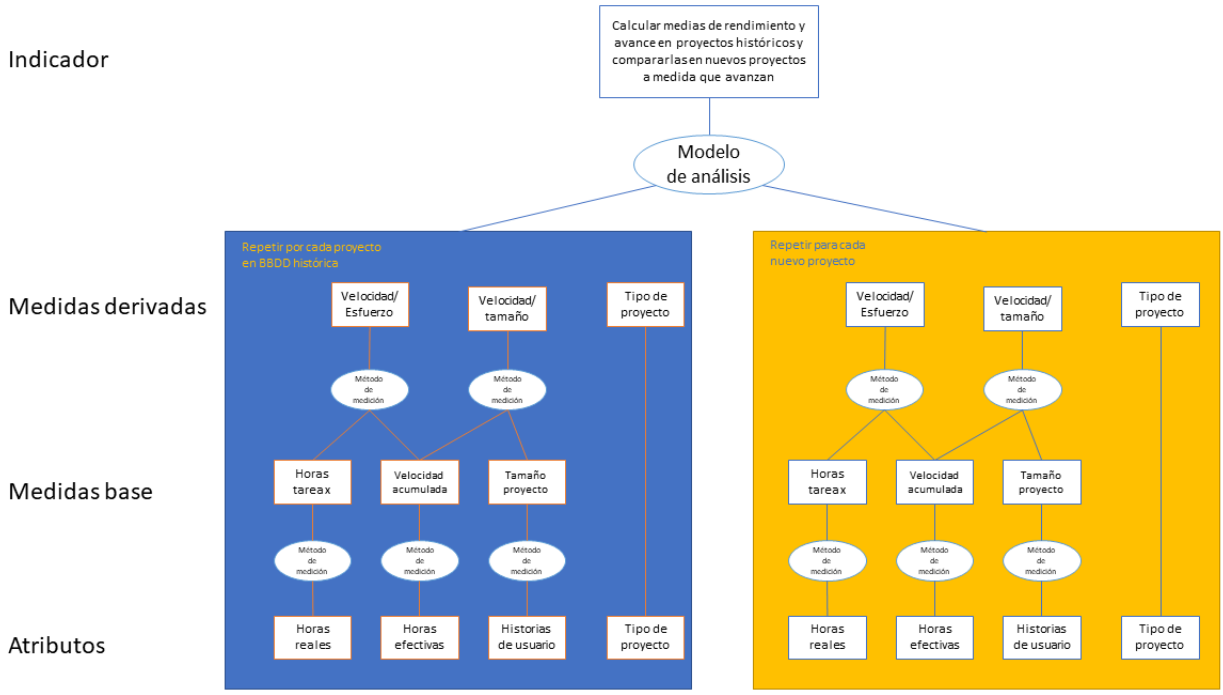
medición, definiremos métricas que proporcionen suficiente visibilidad para satisfacer las necesidades y poder apoyar la toma de decisiones. El detalle de las métricas lo dejamos para el siguiente apartado, si bien parece que las relacionadas con el gráfico *burn-down* son las adecuadas.
También el plan recogerá cuando se realizan y dado que hablamos de un seguimiento del avance, propondremos que se tomen diariamente.

A continuación, en paralelo al desarrollo del proyecto, **realizaremos la medición**. Mediremos, recopilaremos diariamente, analizaremos los resultados diariamente y los presentaremos en la reunión de retroalimentación de Scrum.

Terminado el proyecto, hacemos balance de la medición, evaluamos el proceso y proponemos **mejoras** (de herramienta, de temporalidad, de las medidas usadas, …)

2) Elabore una lista de atributos, medidas base, medidas derivadas y, sobre todo, indicadores a tener en cuenta para responder a las necesidades de información planteadas. **(1,5 punto)**

Para medir el grado de avance de un proyecto y hacer una comparación con datos históricos, vamos a partir de la siguiente hipótesis: Consideramos que el proyecto va bien cuanto mayor sea la ratio horas efectivas / horas reales. Para ello usaremos la velocidad efectiva (M3) y el esfuerzo real medido (M8). En todo momento diferenciaremos según la tipología de proyecto (M4) ya que se observa una clara segmentación en los datos. Las ratios por encima de la media indicarán que nuestro proyecto avanza adecuadamente. También serían validas soluciones que se centraran en la velocidad y/o el valor entregado.



Examen de teoría/problemas

Ejercicio 2. Estimación. (1.5 puntos)

A partir de los datos históricos de proyectos de la tabla siguiente,

3) construya un método paramétrico/algorítmico personalizado para la empresa que a partir de una estimación de tamaño obtenga el esfuerzo necesario:

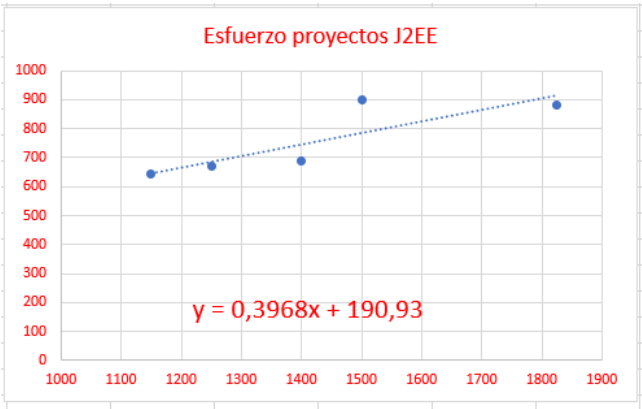
Proyecto id	Tamaño estimado en puntos-historia	Sprints realizados	Velocidad	Arquitectura	Complejidad ciclomática	WMC (weighted methods per class)	Defectos	Esfuerzo
86	1250	3	400	J2EE	17	95	7	670,2
127	890	2	500	Web App	11	60	3	529,7
186	660	2	450	Web App	12	55	4	499,0
242	900	2	470	Web App	10	60	3	511,7
246	1500	4	430	J2EE	15	90	5	899,5
311	1150	3	380	J2EE	14	110	7	643,7
336	1400	3	415	J2EE	16	100	6	687,9
341	850	2	440	Web App	10	50	2	463,0
345	1825	4	420	J2EE	15	85	4	880,4

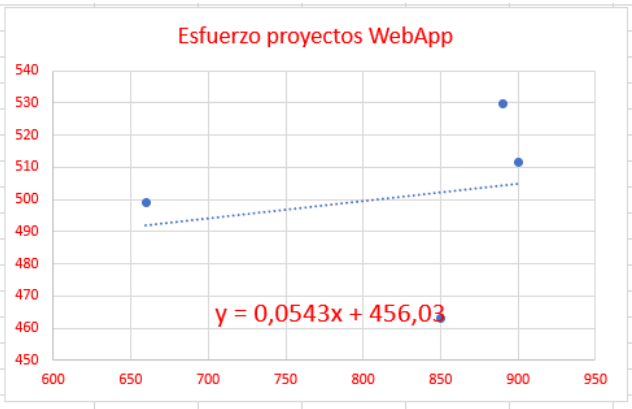
Tabla 1. Datos históricos de proyectos.

Nota: Tenga en cuenta que consideramos que la arquitectura influye de manera clara y directa cuantitativamente en el esfuerzo.

El método paramétrico/algorítmico más adecuado sería aquel que se base en nuestros datos históricos y en la correlación directa entre tamaño y esfuerzo. De entre los métodos que conocemos parece que el más adecuado es, a partir de los datos históricos y aplicando regresión lineal, obtener una fórmula que correlacione y permita estimar esfuerzo a partir del tamaño.

Observamos claramente que los datos se agrupan en dos arquitecturas. Como la nota nos indica la influencia de la arquitectura en la determinación cuantitativa del esfuerzo, vamos a optar por obtener dos fórmulas, una para cada arquitectura.





Ejercicio 3. Estimación funcional de la web de la UHU. (1,5 puntos)

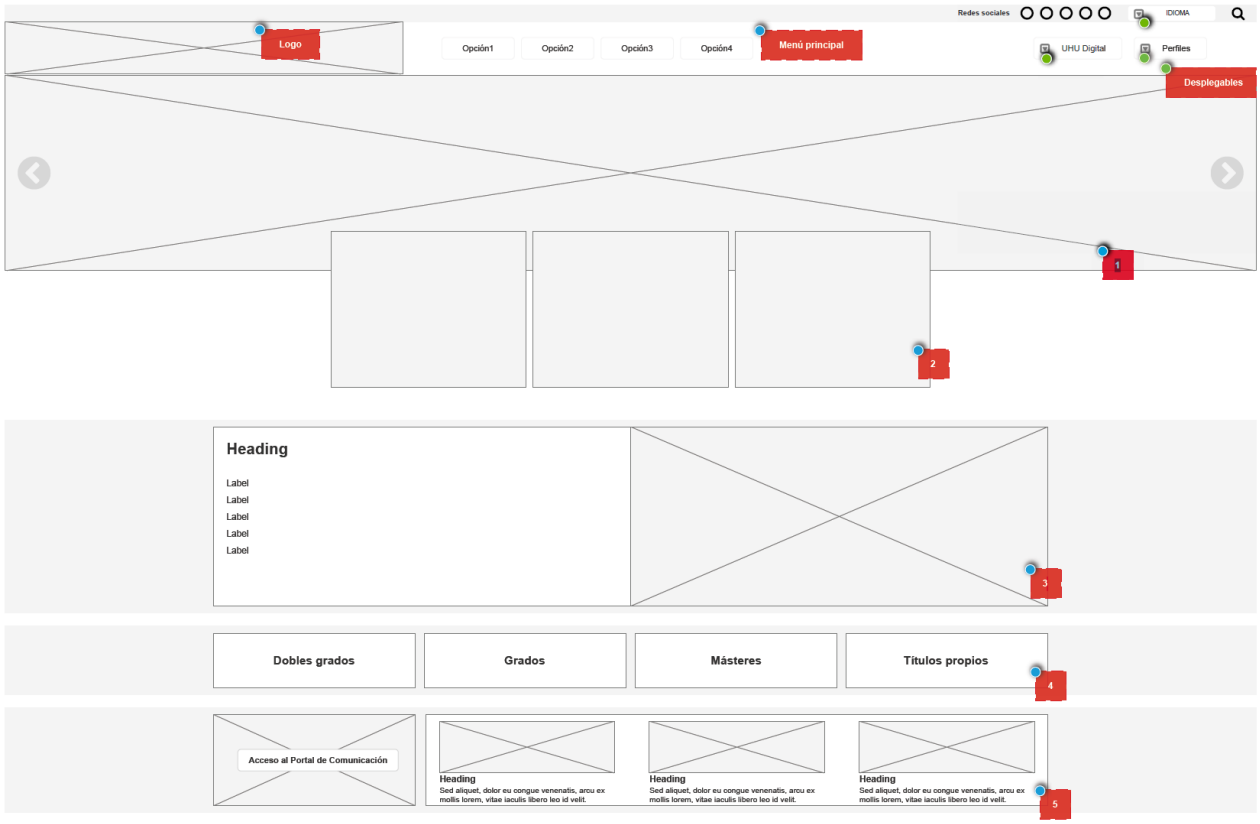


Ilustración 1. Wireframe de la página principal de la nueva web

La Universidad va a poner en marcha una nueva web que se basa en un gestor de contenidos. El gestor almacena internamente la estructura de la web y los componentes incrustados dividiendo el sitio en páginas, las páginas en bloques (que se repiten en todas las páginas) y los bloques en componentes.

La estructura de la página principal de la web incluye:

- Bloque de elementos superiores:
  - Logo de la UHU, arriba a la izqda.
  - Menú principal, con cuatro listas despegables de elementos con enlaces (5 cada uno).
  - Redes sociales
  - UHU digital (desplegable con enlaces a 10 apps)
  - Perfiles (desplegable con enlaces a 6 páginas)
- Bloque de presentación
  - Carrusel con 3 imágenes y enlaces. Etiquetado en ilustración con número 1.
  - Enlaces destacados (3 imágenes y enlaces). Etiquetado en ilustración con número 2.
- Bloques de contenidos: Compuesto en este caso por 3 bloques:
  - Bloque dinámico de contenidos (3 elementos en carrusel, cada elemento consta de un texto html y una imagen). Etiquetado en ilustración con número 3.
  - Bloque estático de contenidos (4 enlaces a páginas intermedias). Etiquetado en ilustración con número 4.
  - Bloque dinámico de contenidos: 1 imagen con enlace a portal de comunicación y 5 noticias (recuperadas del portal de comunicación, donde se crean) que se muestran en carrusel, cada una tiene una imagen, un enlace y un texto breve. Etiquetado en ilustración con número 5.

Además, usa una cookie con 6 datos para almacenar las preferencias del usuario.

Antes de contratar el desarrollo de la nueva web se realizó una estimación del tamaño funcional del proyecto aplicando métricas de Punto Función.

4) Identifique qué tipo es, **justificando su respuesta**, la transacción '*portada de la web*'. **(0,25 ptos.)**

Se trata de una funcionalidad que, a partir de información almacenada internamente, la manipula visual y estéticamente para proporcionar dinámicamente una página web, por lo que la consideramos una 'salida externa'.

5) Tenga en cuenta no sólo la transacción, sino también todos los Grupos Lógicos de Datos y datos elementales implicados que aparecen claramente especificados en el sistema y que habrán de recogerse en la tabla de puntos-función no ajustados. **(0,5 ptos.)**

Se nos dice que se divide el sitio en páginas, las páginas en bloques (que se repiten en todas las páginas) y los bloques en componentes.
Como el bloque es el elemento que se repite, va a ser el centro de nuestra estructura de almacenamiento. Cada página va a tener un bloque superior, un bloque de presentación y un bloque de contenidos (3 ficheros fuente). Además, las cookies almacenan en el propio equipo las preferencias del usuario, por lo que son otro grupo de datos. A todos los consideramos internos porque no se hace referencia a ningún otro sistema. El portal de comunicación forma parte de la web. Si alguien lo considera externo, también se valora.

Transacción/Grupo Lógico	Tipo de Componente (EE, SE, GLDI, GLDIZ Y CE)	Número de ficheros y datos elementales	Lista de datos elementales
Portada de la web	SE	F=4, DE=42+12+26+6=86	
Bloque superior	GLDI	F=1, DE=1+4·5+5+10+6=42	
Bloque presentación	GLDI	F=1, DE=6+6=12	
Bloque contenidos	GLDI	F=1, DE=6+4+15+1=26	
cookies	GLDI	F=1, D=6	

Rellene la tabla de cálculo del total de puntos-función no ajustados:

Descripción	Sencilla	Media	Compleja	Total PF
Nº Entradas externas	x 3	x 4	x 6	
Nº Salidas externas	x 4	x 5	**1** x 7	**7**
Nº Grupos Lógicos de Datos Internos	**4** x 7	x 10	x 15	28
Nº Grupos Lógicos de Datos de Interfaz	x 5	x 7	x 10	
Nº de Consultas Externas	x 3	x 4	x 6	
Total Puntos Función No Ajustados				**35**

**Las tablas de cálculo de complejidad se encuentran en los anexos del examen*

Se consideran que las características particulares de este proyecto son ligeramente complejas, por lo que la influencia de los requisitos no funcionales y de las restricciones técnicas será escasa (ajuste negativo del 10%).

6) Calcule el tamaño funcional necesario en la aplicación original en Punto-Función Ajustado. **(0,25 puntos)**

Se nos dice que el ajuste, aunque escaso, es negativo. Eso quiere decir que penaliza/aumenta el tamaño/esfuerzo:
PFA=PFNA*FA=35*(1+0,10)=38.5 PF

7) Teniendo en cuenta la arquitectura, ¿qué esfuerzo requerirá el desarrollo de la aplicación propuesta? Use para ello la función obtenida mediante el método algorítmico en el ejercicio 2, teniendo en cuenta que 12 puntos-historia equivalen a 1 punto-función.

(0,5 puntos)

De las dos ecuaciones obtenidas en el ejercicio anterior, tomamos la de **aplicaciones web**:

Y=0,0543x+456,03; E=0,0543 (tamaño en puntos historia)+456,03
tamaño en puntos historia = 38.5 PF * 12 puntos-historia / PF = 462
E = 0,0543 (462) + 456,03 = 481,11 (no indicamos unidad porque la tabla 1 no la recoge)

Ejercicio 4. Complejidad ciclomática (2 puntos)

El *Software Engineering Institute* (SEI) presento la siguiente tabla de riegos en su artículo *Software Technology Reference Guide*:

Complejidad Ciclomática	Riesgo
1-10	Programa simple, sin mucho riesgo
11-20	Programa medianamente complejo, riesgo moderado
21-50	Programa complejo, alto riesgo
> 50	Programa no testeable, riesgo muy alto

Complejidad ciclomática – tabla de riesgos

Formas parte de un equipo de calidad que ha fijado el siguiente indicador:
"aquellos procedimientos y métodos con riesgo moderado requieren pruebas de caminos linealmente independientes".

Dada la clase que aparece en el **anexo II**, se le pide que:

8) determine si cumple el indicador y actúe en consecuencia. **(1 punto)**

Para calcular la complejidad ciclomática tenemos 3 métodos, 2 topológicos (a partir de la construcción de un grafo de control y un tercero en base al número de condicionales.

El número de condiciones nos determina una V(G) entre 11 y 20, por tanto, el programa tiene un riesgo **moderado**.
Atendiendo al indicador, requiere pruebas de caminos linealmente independientes, por lo que hemos de dibujar el grafo (ver última página)

Dado que V(G)= 16, necesitamos un conjunto mínimo de16 caminos de prueba:

Camino 1: **0-1-2-29-33**
Camino 2: 0-1-2-29-**30**-33
Camino 3: 0-1-2-29-30-**31**-33
Camino 4: 0-1-2-**3-4**-2-…
Camino 5: 0-1-2-3-**5**-7-**15-16**-2-…
Camino 6: 0-1-2-3-5-**6**-7-…
Camino 7: 0-1-2-3-5-7-**13-14**-2-…
Camino 8: 0-1-2-3-5-7-**11-12**-2-…
Camino 9: 0-1-2-3-5-7-**17-18**-2-…

Camino 10: 0-1-2-3-5-7-**8-9-19-20-21-28-**2-…
Camino 11: 0-1-2-3-5-7-8-9-**10**-19-20-21-28-2-…
Camino 12: 0-1-2-3-5-7-8-9-19-**22-23-**28-2-…
Camino 13: 0-1-2-3-5-7-8-9-19-**24-25-**28-2-…
Camino 14: 0-1-2-3-5-7-8-9-19-**26-27-**28-2-…
Camino 15: 0-1-2-29-**30-**33
Camino 16: 0-1-2-29-30-**31-**33

9) Defina la relación entre prueba exhaustiva y complejidad ciclomática. **(0,5 punto)**

Se trata de justificar que la prueba exhaustiva (probar todos los caminos, con todas las posibles combinaciones de orden de ejecución de las sentencias) es imposible por costosa en tiempo y recursos y que la complejidad ciclomática nos ayuda a identificar el conjunto mínimo de pruebas que nos garantiza que todas las sentencias y condiciones han sido recorridas al menos una vez.
Y que esta estrategia es más eficiente que la exhaustividad.

10) Defina la relación entre grafo de McCabe y la facilidad de mantenimiento. **(0,5 punto)**

En esta pregunta debemos justificar que existen trabajos empíricos que han comprobado que aquellos programas que tenían mayor complejidad ciclomática eran susceptibles de contener más errores. Por lo tanto, son dos conceptos que correlacionan matemáticamente y por eso el grafo y la complejidad calculable a partir del grafo es un muy buen indicador de los métodos/clases a los que revisar/probar especialmente.

Ejercicio 5. Medición en SCRUM (2,5 puntos).

Un equipo de desarrollo desea planificar su siguiente proyecto bajo la metodología Scrum. Para ello va a recopilar el 'product backlog', lo va a repartir en los distintos sprints y, por último, va a plantear las entregas de versiones. Como referencia dispone de los datos históricos de la Tabla 1.

La pila de producto ha sido definida de la siguiente manera:

Historia de Usuario	Tarea	Prioridad	Valor aportado	Estimación en PU
H1	T1.1	5	60	45
	T1.2	4	50	45
	T1.3	2	30	30
H2	T2.1	5	70	45
	T2.2	4	30	25
	T2.3	4	30	15
	T2.4	2	10	20
	T2.5	2	10	25
H3	T3.1	5	50	30
	T3.2	4	40	20
	T3.3	3	40	10
	T3.4	2	30	40

El usuario nos indica que quieren que se le entreguen historias de usuario cuando se acaben pero que el criterio de selección de tareas es exclusivamente 80% prioridad y 20% valor aportado.

Realice un **diagrama Burn-up** que indique:

11) Qué tareas se realizan en cada sprint (teniendo en cuenta prioridad, valor aportado y velocidad). **(0,75 puntos)**

El primer dato que necesitamos para poder seleccionar qué tareas de la pila de producto pasan a la pila del sprint es conocer la velocidad.
No se nos indica por lo que tendremos que calcularla a partir de los datos históricos.

Como no se nos indica la tipología del proyecto, calculamos la media de la velocidad a partir de la tabla 1.
Media (400, 500, 450, 470, 430, 380, 415, 440, 420)=433,88

Lo siguiente sería ordenar tareas por el criterio de selección:

Historia de Usuario	Tarea	Prioridad	Valor aportado	Estimación en PU	Criterio	Orden
	T1.1	5	60	45	16,00	2
	T1.2	4	50	45	13,20	4
H1	T1.3	2	30	30	7,60	10
	T2.1	5	70	45	18,00	1
	T2.2	4	30	25	9,20	7
	T2.3	4	30	15	9,20	8
H2	T2.4	2	10	20	3,60	12
	T2.5	2	10	25	3,60	11
	T3.1	5	50	30	14,00	3
	T3.2	4	40	20	11,20	5
	T3.3	3	40	10	10,40	6
H3	T3.4	2	30	40	7,60	9

Ahora repartiríamos las tareas entre los esprints. Suponemos que la unidad de medida, por coherencia, son los puntos·usuario (PU).
La suma de las tareas supone 350 PUs.
Dado que la velocidad estimada es de 433,88 PUs/sprint podemos concluir que **todas las tareas se realizan en el primer sprint en el orden que aparece en la última columna de la tabla**.

12) Cuantas versiones y en qué momento (semana) se entregarán. **(0,75 puntos)**

En las metodologías ágiles, los esprints tienen una duración fija, normalmente entre 2 y 4 semanas, que favorece la creación de un ritmo permanente de entrega de software. Como vimos en el uso de los tableros Kanban, el desarrollo por esprints puede evolucionar fácilmente a un entorno de entrega continua.

Dado que no se nos indica en el enunciado, presuponemos una duración de 4 semanas. Si, a la vez, partimos de un enfoque en el que las entregas son al finalizar el sprint, obviamente se entregaría una sola versión. ¿Cuándo?

350 $_{PUs}$ / 433,88 $_{PUs/sprint}$ = 0,80 sprint -> semana 0,80 * 4 semanas/sprint = 3,22
Se entregaría el segundo día de la tercera semana.

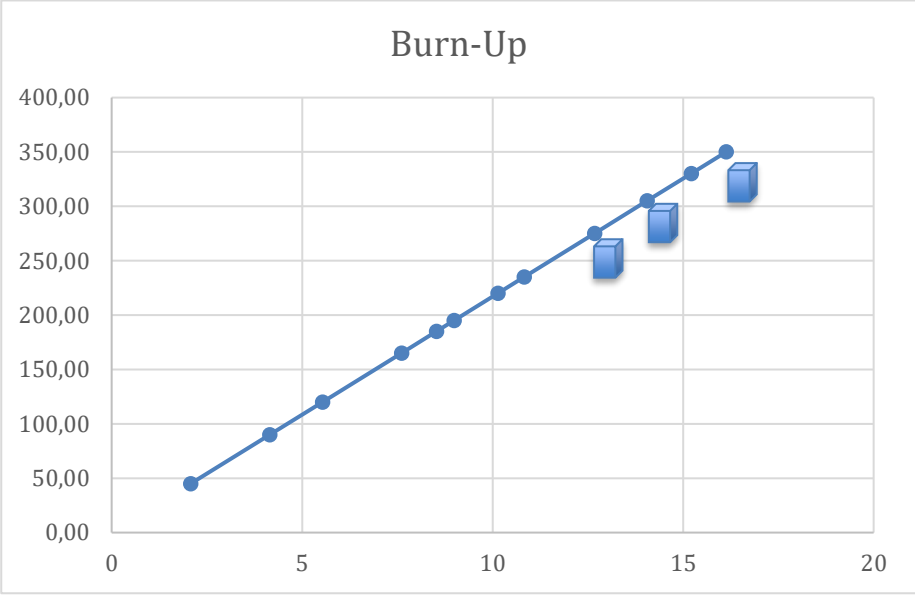
Velocidad = 433,88
Velocidad diaria: 433,88 / (4 semanas/sprint * 5 días/semana) = 21,7

En un gráfico Burn-up quedaría así:

Si, por el contrario, optamos por un enfoque de entrega continua, en línea con DevOps, podemos realizar los siguientes cálculos:

Siguiendo el orden prefijado en apartado anterior:

Historia de Usuario	Tarea	Estimación en PU	Orden	Acum.
H1	T1.1	45	2	90
	T1.2	45	4	165
	T1.3	30	10	305
H2	T2.1	45	1	45
	T2.2	25	7	220
	T2.3	15	8	235
	T2.4	20	12	350
	T2.5	25	11	330
H3	T3.1	30	3	120
	T3.2	20	5	185
	T3.3	10	6	195
	T3.4	40	9	275

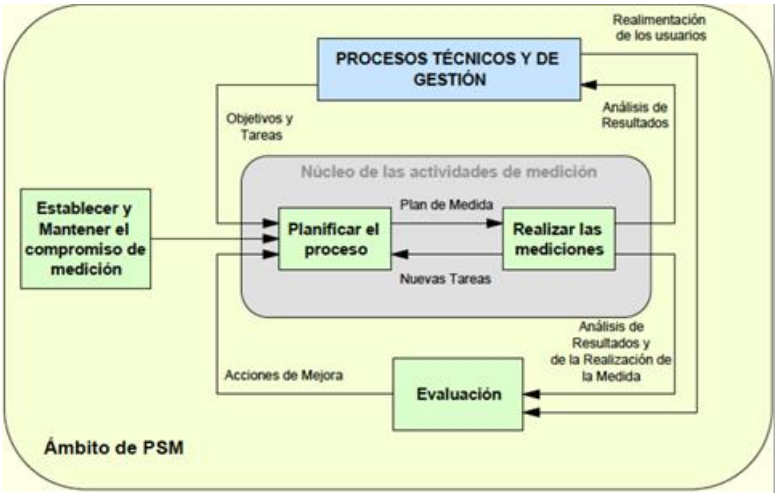


H1, entrega 4 marzo
H2, entrega 8 marzo
H3, entrega 2 marzo

13)Gráfico Burn-down. Funcionamiento y utilidad. **(1 punto)**

Explicar cómo se construye y su utilidad para el seguimiento del proyecto.

Anexo I. Fórmulas y tablas para el examen.



FA = 0.65 + (0.01 * SVA)
PFA = PFNA * FA

Ficheros o Grupos Lógicos de Datos Internos

Tipos de Registros	Tipos de datos elementales: 1 a 19	20 a 50	51 ó más
1	S	S	M
2 a 5	S	M	C
6 ó más	M	C	C

Ficheros o Grupos Lógicos de Datos Externos

Tipos de Registros	Tipos de datos elementales: 1 a 19	20 a 50	51 ó más
1	S	S	M
2 a 5	S	M	C
6 ó más	M	C	C

Entradas externas

Ficheros Referenciados	Tipos de datos elementales: 1 a 4	5 a 15	16 ó más
0 ó 1	S	S	M
2	S	M	C
3 ó más	M	C	C

Salidas externas

Ficheros Referenciados	Tipos de datos elementales: 1 a 5	6 a 19	20 ó más
0 ó 1	S	S	M
2 ó 3	S	M	C
4 ó más	M	C	C

Fórmula de la regresión lineal simple: y=a+b·x:

$$b = \frac{\sum (x_i - m_x)(y_i - m_y)}{\sum (x_i - m_x)^2}$$

$$a = m_y - bm_x$$

Métodos algorítmicos clásicos

$E = 5.2 \cdot (KLDC)^{0.91}$ **Modelo de Walston-Felix**

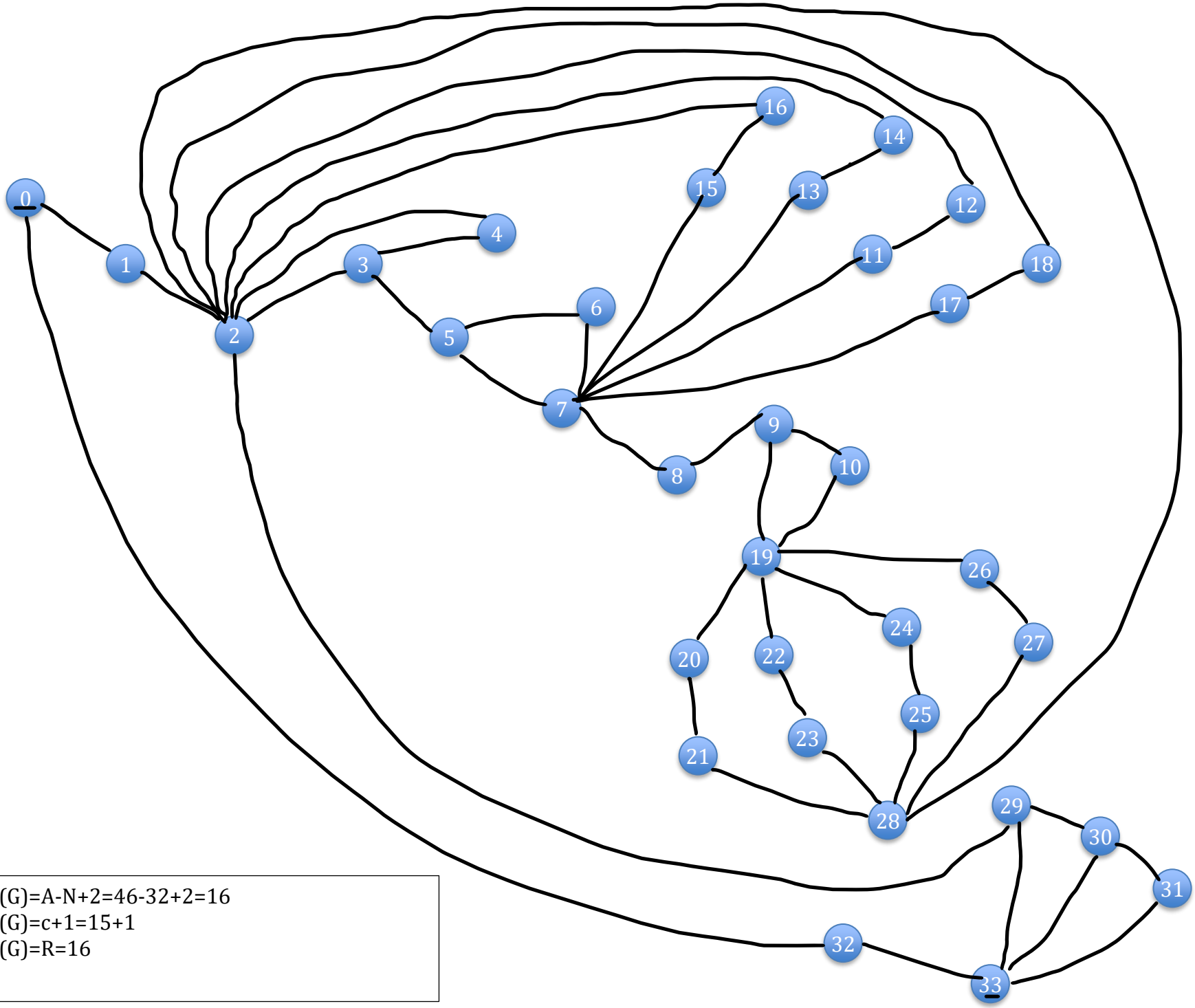
$E = 5.5+0.73 \cdot (KLDC)^{1.16}$ **Modelo de Bailey-Basili**

$E = 3.2 \cdot (KLDC)^{1.05}$ **Modelo simple de Boehm**

$E= 5.288 \cdot (KLDC)^{1.047}$ **Modelo Doty para KLDC >9**

Anexo II.

```
package solrparser;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import org.apache.solr.common.SolrInputDocument;

public class SolrDocParser {

    public static ArrayList<SolrInputDocument> parseDocs(final String path) {
        ArrayList<SolrInputDocument> docs = new ArrayList<>();

        try {
            BufferedReader br = new BufferedReader(new FileReader(path));

            String line;
            SolrInputDocument doc = null;

            // Bandera para indicar el estado en el que se encuentra
            // el documento actual. Como si fuera una maquina de estados.
            /**
             * Estado 0 - Sin inicializar
             * Estado 1 - Inicializado
             * Estado 2 - Agregando titulo
             * Estado 3 - Agregando autor
             * Estado 4 - Agregando texto
             * Estado 5 - Obviando lineas
             */
            int state = 0;

            // Para el texto del titulo y descripcion
            StringBuilder tempTitle = new StringBuilder();
            StringBuilder tempText = new StringBuilder();

            while ((line = br.readLine()) != null) {
                if (line.le                ) == 0) contin   // Linea en blanco

                if (line.length()  >
                    String init = l      substring(0,2

                switch (init) {
                    case ".I": {
                        if (doc !=  l) {
                            doc.add     d("title",
remove_incorrect_whitespaces(tempTitle.toString()));
                            doc.addField("text_en",
remove_incorrect_whitespaces(tempText.toString()));
                            docs.add(doc);
                        }

                        doc = new SolrInputDocument();
                        tempTitle = new StringBuilder();
                        tempText = new StringBuilder();

                        doc.addField("id",
                                line.replaceAll(" ", "").split("I")[1]);
                        state = 1;
                        break;
                    }
                    case ".T":
                        state = 2;
                        continue;
                    case ".A":
                        state = 3;
                        continue;
                    case ".W":
                        state = 4;
                        continue;
                    case ".X":
                        state = 5;
                        continue;
                }
            }

            switch (state) {
                case 2: // Agregar    l titulo
                    tempTitle.app      ine).append(" ");
                    break;
                case 3: // Agregar    l autor
                    doc.addField("      or", line);
                    break;
                case 4: // Leyendo    texto de descripcion
                    tempText.appe      ne).append(" ");
                    break;
                case 5: // Obviand    neas
                    break;
            }
        }

        if (line == null && doc != null) {
            doc.a      ield("title",
                    remove_inco      _whitespaces(tempTitle.toString()));
            doc.a      ield("text_e
                    remove_incorrect_whitespaces      Text.toString()));
            docs.add(doc);
        }
    } catch (IOE      ion ex) {
        System.o      rintln("An error occurred!");
        ex.printStackTrace();
    }

    retur      s;
}

// Método auxiliar para eliminar los espacios al principio
// final de un String
private static String remove_incorrect_whitespaces(final String base) {
    return base.replaceAll("^\\s+", "").replaceAll("\\s+$", "");
}

}
```



V(G)=A-N+2=46-32+2=16
V(G)=c+1=15+1
V(G)=R=16