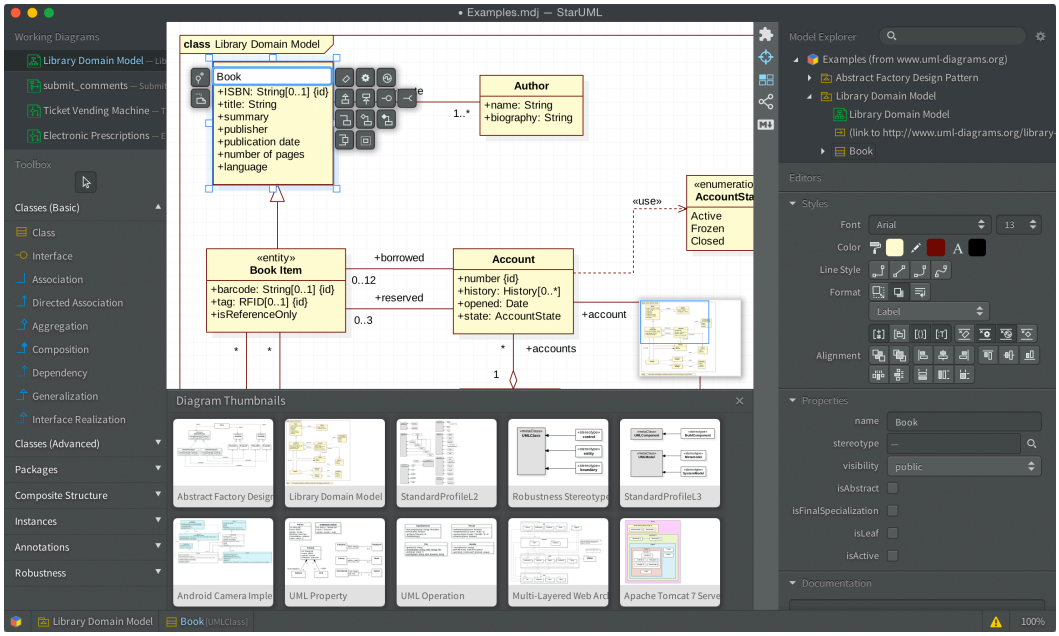
	Calidad, Medición y Estimación de Producto y Proceso Software	4º Grado en Ingeniería Informática Itinerario de Ingeniería del Software
	Examen Final de Teoría	7/2/2017



Ejercicio 1: Análisis de Punto Función con IFPUG (2 puntos)

Una vez finalizada y **desplegada** la versión 1.0 de nuestro software de modelado UML, empezamos a planificar diversas modificaciones, tras recibir el feedback de los usuarios. Para estimar el esfuerzo necesario se parte del siguiente cuadro, usado en la estimación de esfuerzo de la aplicación original.

Descripción	Sencilla	Media	Compleja	Total PF	
Nº Entradas externas	6 x 3	2 x 4	2 x 6 /1	38	6
Nº Salidas externas	4 x 4	2 x 5	2 x 7 /1	40	7
Nº Grupos Lógicos de Datos Internos	8 x 7	2 x 10 /2	4 x 15	136	20
Nº Grupos Lógicos de Datos de Interfaz	0 x 5	2 x 7	0 x 10 /1	14	10
Nº de Consultas Externas	6 x 3	3 x 4	1 x 6	36	
Total Puntos Función No Ajustados				264	43

Las **modificaciones** planteadas, según nuestros analistas, van a suponer el **desarrollo de 2 nuevas transacciones**:

-**Generación de código**: A partir de plantillas de diferentes lenguajes, se genera el esqueleto de las clases, con cabeceras, atributos y definiciones de métodos.

-**Edición de plantillas**: Se pueden añadir/modificar plantillas que permitan la generación de código en nuevos lenguajes.

En ambos casos, la complejidad de la transacción ha sido analizada como “alta”. Estas nuevas funcionalidades han supuesto la creación y acceso a **dos** tablas

nuevas en la Base de Datos propia de la aplicación (ponderados con complejidad media) y el acceso vía API a **un** registro gestionado por otra aplicación (ponderado con complejidad alta).

Dado que se trata de una modificación, los aspectos no funcionales han sido debidamente tratados en el desarrollo inicial, por lo que se considera que las 14 características generales del sistema en las que se basa el factor de ajuste tienen una influencia mínima (10% de su valor máximo):

a) Calcule el tamaño funcional necesario ~~en la aplicación original~~ en Punto-Función Ajustado, así como en la **modificación** planteada. (1 punto)

```
FA = 0.65+(0.01*SVA)
     FA entre 0,65 y 1,35
PFA = PFNA * FA
```

El proyecto original ha sido ya desplegado, por lo que tenemos que planificar y estimar sólo la modificación planteada.
Cada característica toma un valor entre 0 y 5, y su suma entre 0 y 70
SVA=10%(14·5)=10%(70)=7; FA=0,65+(0.01*7)=0,72
$PFA_{modif.}$=43·0,72=30,96 PF

b) Si disponemos de 2.5 trabajadores disponibles, ¿cuántos días requerirá el desarrollo de la **modificación** propuesta? Use para ello las tablas ISBSG, sabiendo que el proyecto se desarrolla en Java para PC.

(1 punto)

	Características	C	E
1	MF	49,02	0,736
2	MR	78,88	0,646
3	PC	48,90	0,661
4	Multi	16,01	0,865
5	3GL	54,65	0,717
6	4GL	29,50	0,752
7	GenAp	68,11	0,660
8	Mantenimiento	52,58	0,683
9	Nuevo	39,05	0,731
10	MF-3GL	65,37	0,705
11	MF-4GL	52,09	0,640
12	MF-GenAp	65,68	0,692
13	MR-3GL	126,3	0,565
14	MR-4GL	62,35	0,694
15	PC-3GL	60,46	0,648
16	PC-4GL	36,48	0,694
17	Multi-3GL	19,82	0,666
18	Multi-4GL	6,49	0,983
19	MF-3GL-Mantenimiento	83,27	0,650

$$Esfuerzo = C \cdot Tamaño^{E}$$

Por encima de cualquier factor, el proyecto es una modificación, por tanto consideramos la categoría **mantenimiento**. Otra opción sería PC-3GL
C=52,58 y E=0,683
Esfuerzo=52,58·30,96^{0,683}=548,324 horas
En PM=548,324/160hpm=3,42PM
Duración=(548,324/2.5)/8=27,416 días laborables

Pregunta 2. Medidas, métricas e identificador. (2 puntos)

Defina de manera breve y clara los siguiente conceptos:

- Medida base o directa,
- Medida derivada o indirecta,
- Indicador

Use como ejemplos para la explicación, elementos del ejercicio 1.
¿Qué es PSM y qué papel juegan esos tres conceptos definidos en PSM?

- Métrica base (directa), **medida de un atributo** que no depende de ninguna otra y cuya forma de medir es un método de medición. P.ej.: LOC, HPD (horas-

programador-dia), CHP (coste por hora-programador). En ejercicio 1, número de ficheros implicados en una transacción, número de datos elementales.

- Métrica derivada (indirecta), **medida que se basa en otra** y mide usando una función de cálculo. P.ej.: HPT (Horas-Programador-Total=ΣHPD), CTP (Coste Total Proyecto=Coste unitario hora*total horas empleadas), Earned value, densidad de defectos, cobertura de las pruebas. En el ejercicio 1, podíamos considerar como derivadas el ajuste de PFNA.
- Indicador, Una medida que es derivada de otras medidas utilizando un **modelo de análisis como forma de medir**. PROD (productividad de los programadores), CAR (carestía del proyecto). En el ejercicio 1, podíamos considerar como indicador el cálculo de la complejidad de una transacción.

Un programa de medición debe proporcionar, al final del proceso, información útil para la toma de decisiones. PSM incorpora un modelo de información que relaciona las entidades que son medidas con las medidas definidas y con las necesidades de información que se satisfacen, el constructor de medición.

El Constructor de la medición describe cómo atributos relevantes de productos y procesos se miden, se combinan y se convierten en indicadores, relacionando atributos y necesidades de información para la toma de decisiones.

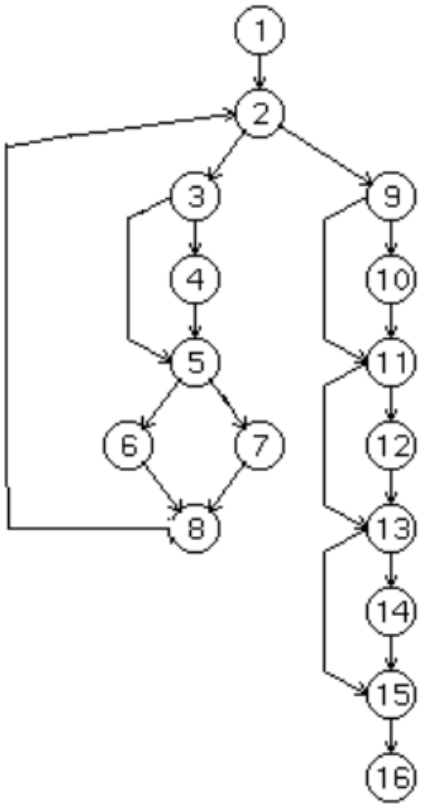
Ejercicio 3: Complejidad ciclomática: McCabe (2 puntos)

Formas parte de un equipo de calidad que ha fijado el siguiente indicador: *"aquellos procedimientos y métodos con complejidad ciclomática > 6 requieren pruebas de caminos linealmente independientes"*.

Dada la siguiente función, determine si cumple el indicador y, en su caso, determine el número de pruebas y concrete los caminos a probar.

```
stcode getlist(char *lin, int *i, stcode *status)
{
 int num, done;
 line2 = 0;
 nlines = 0;
 done = (getone(lin,i,&num,status)!=OK);
 while (!done)
 {
   line1 = line2;
   line2 = num;
   nlines++;
   if (lin[*i]==SEMICOL)
      curln = num;
   if ((lin[*i]==COMMA)||(lin[*i]==SEMICOL))
      {
      *i = *i + 1;
      done = (getone(lin,i,&num,status)!=OK);
      }
   else
      done = 1;
 }
nlines = min(nlines,2);
if (nlines == 0)
      line2 = curln;
if (nlines <= 1)
      line1 = line2;
if (*status != ERR)
      *status = OK;
return(*status);
}
```

Dado que nos piden concretar, llegado el caso, los caminos a probar, optamos por dibujar el grafo de control de la función.




```
stcode getlist(char *lin, int *i, stcode *status)
/*  1 */ {
/*  1 */  int num, done;
/*  1 */  line2 = 0;
/*  1 */  nlines = 0;
/*  1 */  done = (getone(lin,i,&num,status)!=OK);
/*  2 */  while (!done)
/*  3 */  {
/*  3 */    line1 = line2;
/*  3 */    line2 = num;
/*  3 */    nlines++;
/*  3 */    if (lin[*i]==SEMICOL)
/*  4 */      curln = num;
/*  5 */    if ((lin[*i]==COMMA)||(lin[*i]==SEMICOL))
/*  6 */    {
/*  6 */      *i = *i + 1;
/*  6 */      done = (getone(lin,i,&num,status)!=OK);
/*  6 */    }
/*  7 */     else
/*  7 */       done = 1;
/*  8 */  }
/*  9 */  nlines = min(nlines,2);
/*  9 */  if (nlines == 0)
/* 10 */    line2 = curln;
/* 11 */  if (nlines <= 1)
/* 12 */    line1 = line2;
/* 13 */  if (*status != ERR)
/* 14 */    *status = OK;
/* 15 */  return(*status);
/* 16 */ }
```

V(G)=R=Número de regiones= 7
V(G)=c+1= 6+1=7
V(G)=A-N+2=21-16+2=7

En nodo 5, se considera adecuado también evaluar por separado la segunda condición, por tanto en ese caso **V(G)=8**

Según el indicador, la complejidad de esta función requiere que se realicen pruebas de caminos linealmente independientes. V(G) nos da el número mínimo de caminos a probar.

Caminos:

1. 1-2-9-10-11-12-13-14-15-16
2. 1-2-9-10-11-12-13-15-16
3. 1-2-9-10-11-13-14-15-16
4. 1-2-9-11-12-13-14-15-16
5. 1-2-3-4-5-6-8-2-9…
6. 1-2-3-5-6-8-2-9…
7. 1-2-3-5-7-8-2-9…

Ejercicio 4. Regresión Lineal (2,5 puntos).

La siguiente tabla recoge datos históricos de proyectos de nuestra organización, en concreto tamaño y esfuerzo requerido.

Proyecto	A	B	C	D	E
Esfuerzo (persona·horas)	525	510	305	700	450
Tamaño (PF)	30	29	15	42	25

Se pide:

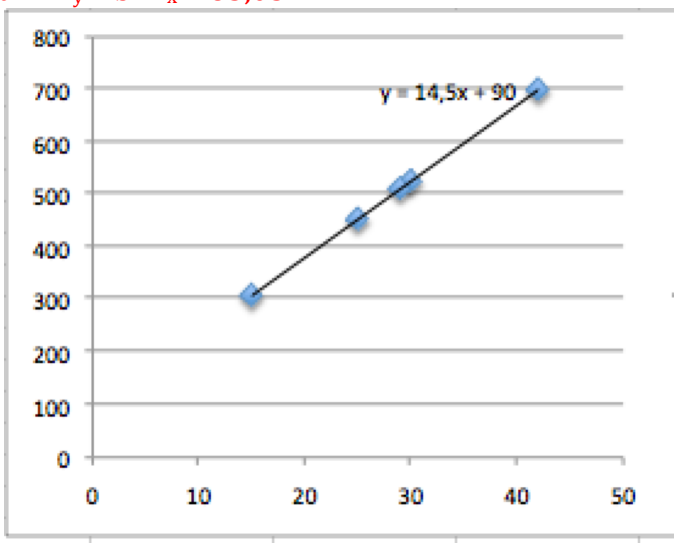
a) Obtenga la formula de la recta que relaciona *Tamaño* y *Esfuerzo* aplicando regresión lineal simple, sabiendo que los coeficientes de la recta se obtienen a partir de las siguientes fórmulas: (1,5 puntos)

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

$$a = m_y - bm_x$$

b = 14,64

$a = m_y - b \cdot m_x = 85,05$



luego la fórmula de la recta que relaciona tamaño y esfuerzo sería:

y (esfuerzo) = a + b·x (-r) = 85,05 + 14,64·x (tamaño)

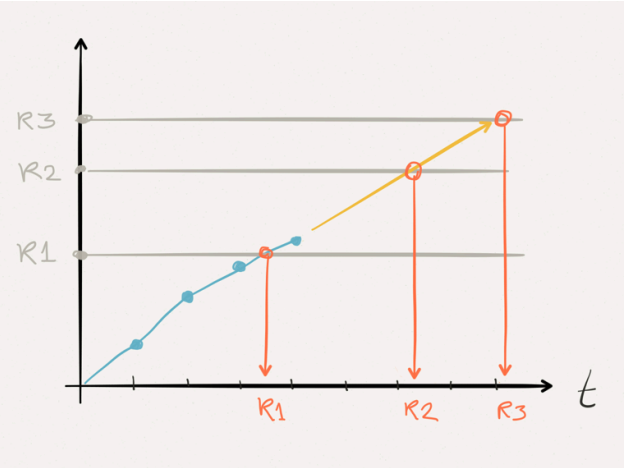
b) Aplicando dicha recta, ¿Cuál será el esfuerzo asociado a la modificación planteada en el **ejercicio 1**? (1 punto)

Tamaño de la modificación en PF = 30,96 PF

Esfuerzo = 85,05 + 14,64·(30,96) = **538,30 Pers.hora**

En PM, E=538,30/160=3,36 Personas·mes

Ejercicio 4. Gráficos Scrum (2 puntos).



Nos encontramos al principio de un proyecto que va a seguir la metodología Scrum. Para ello, reunidos con el *Propietario del Producto*, estamos intentando concretar el *Plan de Producto*.

La *Pila de Producto* ha sido definida de la siguiente manera:

Historia de Usuario	Tarea	Prioridad	Estimación en Puntos de Usuario
H1	T1.1	5	90
	T1.2	4	90
	T1.3	2	60
H2	T2.1	5	95
	T2.2	4	55
	T2.3	4	30
	T2.4	2	40
	T2.5	2	50
H3	T3.1	5	60
	T3.2	4	40
	T3.3	3	20
	T3.4	2	80

El usuario nos indica que quieren que se le entreguen historias de usuario acabadas y en el orden en que están en la pila.
Si el **valor aportado** por cada Historia de Usuario es H1 (30), después H2 (20) y finalmente H3 (10),

- la velocidad del equipo de desarrollo es de 150 puntos por sprint y
- la duración de cada sprint es de 2 semanas,

Realice un diagrama Burn-up que indique:

a) Qué tareas se realizan en cada sprint (teniendo en cuenta geométricamente prioridad y valor aportado, así como las limitaciones de la velocidad). Cuantas versiones y en qué momento (semana) se entregarán.

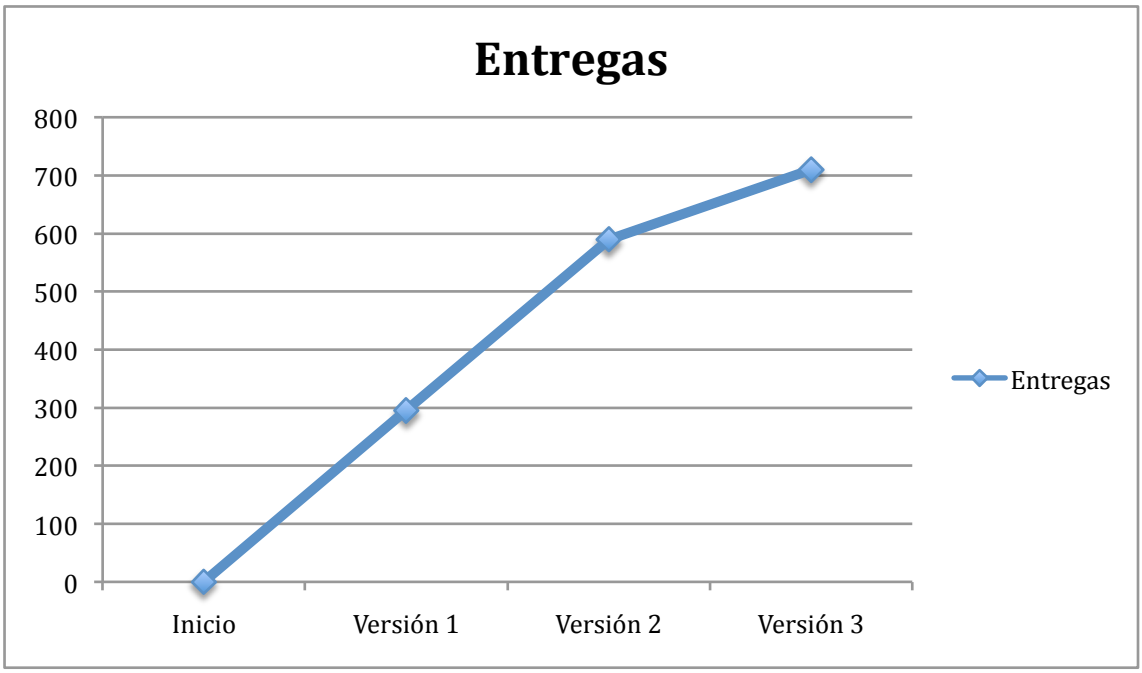
(1,5 puntos)

Nuestro objetivo será entregar lo antes posible Historias de Usuario (que coincidirán con versiones), pero tenemos la limitación impuesta por la velocidad del equipo en la iteración.
Como habrá veces que no podamos incluir en una iteración todas las tareas de una historia de mayor valor, ordenemos tareas atendiendo a prioridad y valor aportado:

Historia de Usuario	Valor	Tarea	Prioridad	V·P	Estimación en Puntos de Usuario
H1	30	T1.1	5	150	90 *
H1	30	T1.2	4	120	90 *
H1	30	T1.3	2	60	60 *
H2	20	T2.1	5	100	95 *
H2	20	T2.2	4	80	55 *
H2	20	T2.3	4	80	30 *
H2	20	T2.4	2	40	40 *
H2	20	T2.5	2	40	50 *
H3	10	T3.1	5	50	60 *
H3	10	T3.2	4	40	40
H3	10	T3.3	3	30	20 *
H3	10	T3.4	2	20	80

Tarea/ Historia	Sprint	Puntos usuario	Puntos iteración Disponibles
T1.1 H1	1	90	150-90=60
T2.2 H2	1	55	60-55=5
T1.2 H1	2	90	150-90=60
T1.3 H1	2	60	0
		V1	Entrega H1
T2.1 H2	3	95	150-95=55
T2.3 H2	3	30	55-30=25
T3.3 H3	3	20	25-20=5
T2.4 H2	4	40	150-40=110
T2.5 H2	4	50	110-50=60
T3.1 H3	4	60	0
		V2	Entrega H2
T3.2 H3	5	40	150-40=110
T3.4 H3	5	80	110-80=30
		V3	Entrega H3

Por lo tanto, se realizarán 5 sprints y 3 entregas, al final de las semanas 4, 8 y 10.



b) Daily Scrum o reunión diaria. Descríbala brevemente. ¿Qué preguntas han de hacerse?

(1 punto)

De manera resumida y esquemática:
Es una reunión diaria, breve (dura 15 minutos), con todos los asistentes atentos y estáticos. Está todo el mundo invitado, si bien sólo pueden hablar los miembros del equipo, Scrummaster y Propietario del producto. No pretende solucionar problemas, es una charla de comunicación y coordinación que pretende evitar otras reuniones innecesarias.
Todos responden a 3 preguntas:

- Qué hiciste ayer?
- Qué vas a hacer hoy?
- Hay obstáculos en tu camino?

de manera que todos se comprometan con el desarrollo del producto.