


| | | |
|---|---|---|
|  | 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 | 6/2/2015 |

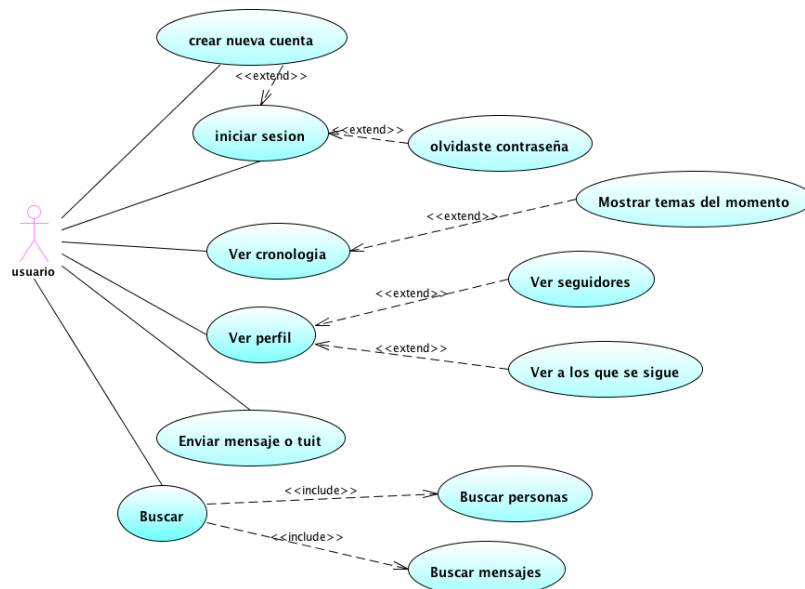
Apellidos y nombre: _____ DNI: _____

Ejercicio 1: ¿Cuánto esfuerzo requiere diseñar Twitter?

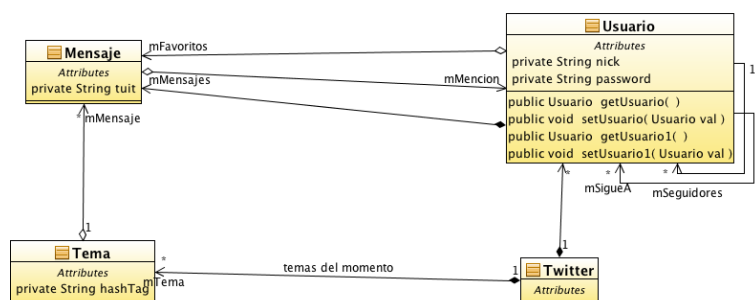


Twitter es bla, bla,

Siguiendo su filosofía, deseamos crear un sistema de mensajería para nuestra empresa, usando un cliente basado en Java Swing. Nuestros analistas han estudiado los requisitos del sistema y han elaborado el correspondiente Diagrama de Casos de Uso.



Así mismo, analizando la información que manipula el sistema, se ha diseñado el siguiente Diagrama de Clases:



Análisis de Punto Función con IFPUG (3 puntos)

Antes de empezar con el desarrollo se inicia un proceso de estimación. En concreto, se desea analizar el tamaño funcional de los Casos de Uso 'Enviar mensaje o tweet' (Botón *Tweet!*) y 'Mostrar temas del momento' (Cuadro de texto *Trending Topics*).



En el CU 'Enviar mensaje o tweet', el usuario introduce una cadena de texto. El sistema analiza si incluye una mención a otro usuario y si incluye algún hashtag, para facilitar la indexación y búsqueda por tema.

Para el CU 'Mostrar temas del momento', cada vez que se actualiza la pantalla se muestran al usuario aquellos temas con mayor número de mensajes asociado en el intervalo de tiempo más reciente.

Tenga en cuenta que la funcionalidad se apoya para sus cálculos en un grupo de *cookies* locales, en los que almacena 14 datos elementales (no dice que los datos elementales se usen en las 2 transacciones).

- a) Identifique de qué tipo son, **justificando su respuesta**, las transacciones 'Enviar mensaje o tweet' y 'Mostrar temas del momento'. Tenga en cuenta no sólo las transacciones, sino también todos los Grupos Lógicos de Datos y datos elementales implicados que aparecen en el sistema y que habrán de recogerse en la tabla de puntos-función no ajustados. **(1 punto)**

| Transacción/Grupo Lógico | Tipo de Componente (EE, SE, GLDI, GLDIZ Y CE) | Número de ficheros y datos elementales | Lista de datos elementales |
|---------------------------|---|--|----------------------------|
| Enviar mensaje o tweet | EE | F: 5 Twitter, usuario, mensaje, [tema] y cookies D: 3 Compl. Media | Nick, texto, hashtag |
| Mostrar temas del momento | SE | F: 3 (Twitter, tema y cookies) D: 1 Compl. Sencilla | hahstag |
| <i>cookies</i> | GLDI | T:1, D: 14 Compl. Sencilla | |
| Twitter | GLDI | T:1, D: 0 Compl. Sencilla | |
| Usuario | GLDI | T:1, D: 2 Compl. Sencilla | |
| Mensaje | GLDI | T:1, D: 1 Compl. Sencilla | |
| tema | GLDI | T:1, D: 1 Compl. Sencilla | |

La transacción 'Enviar tweet' recoge información externa al sistema que introduce por teclado un actor externo y la guarda internamente en nuestro sistema, por lo que claramente es una Entrada Externa. Los ficheros implicados son los necesarios para almacenar y poder recuperar posteriormente ese mensaje, así como el de cookies que se nos comenta en el enunciado.

La transacción 'Mostrar temas' realiza un cálculo sobre información que tenemos registrada en nuestro sistema para mostrarla y difundirla visualmente entre los usuarios, por lo que es claramente una salida externa.

El enunciado nos pide que se analicen todos los grupos lógicos de datos implicados, por lo que habrá que tener en cuenta todas las clases del Diag. de clases y el grupo de cookies.

b) Rellene la tabla de cálculo del total de puntos-función no ajustados: **(0.5 puntos)**

| Descripción | Sencilla | Media | Compleja | Total PF |
|--|--------------|--------------|----------|-----------|
| Nº Entradas externas | x 3 | 1 x 4 | x 6 | 4 |
| Nº Salidas externas | 1 x 4 | x 5 | x 7 | 4 |
| Nº Grupos Lógicos de Datos Internos | 5 x 7 | x 10 | x 15 | 35 |
| 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 | | | | 43 |

**Las tablas de cálculo de complejidad se encuentran en la última hoja del examen*

Por último, indicar que, para el ajuste en base a las características generales del sistema, se nos dice que los factores producen un ajuste favorable del 15%.

c) Calcule el tamaño funcional necesario en la aplicación original en Punto-Función Ajustado. **(0.5 puntos)**

$$FA = 1 - 0,15 = 0,85$$

$$PFA = 43 * 0,85 = 36,55 \text{ PFA}$$

$$FA = 0.65 + (0.01 * SVA)$$

$$PFA = PFNA * FA$$

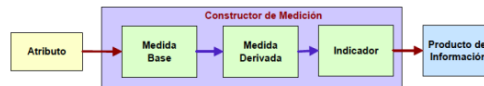
d) Punto-Función. Defina brevemente qué es y cuáles son las ventajas de su uso frente a las Líneas de Código. **(1 punto)**

Es una métrica objetiva para establecer el tamaño y complejidad de Sist. Informáticos basada en la cantidad de funcionalidad requerida y entregada a los usuarios de manera independiente a la tecnología usada.

Ventajas:

- Basado en la perspectiva del usuario
- Independiente del lenguaje, metodologías y herramientas
- Mejor entendimiento por parte de los stakeholders no técnicos
- Ampliamente aceptado
- Benchmarking disponibles
- Existe una organización que lo rige

Ejercicio 2. PSM (2 puntos)



Hemos sido contratados por una Organización de Desarrollo de Software como Ingenieros de Software y una de las primeras tareas que se nos encarga es la realización de un proceso de evaluación y mejora del proceso de desarrollo.

La empresa dispone de las siguientes mediciones por proyecto:

- M1 Control de tiempos de cada fase
- M2 Distribución del tiempo entre cada fase
- M3 Cantidad de Funcionalidad (basada en análisis de los Puntos de Función)
- M4 Esfuerzo. Cantidad de trabajo en Personas/Mes.
- M5 Fiabilidad. Expresada en ratio de defectos.
- M6 Productividad (expresada en horas por PF) = Esfuerzo / PF
- M7 Tiempo / Calendario. Duración del proyecto.
- M8 Velocidad de entrega (expresada en PF por mes) = PF / Duración

Para mejorar el proceso de desarrollo consideramos que una buena estrategia podría ser considerar aquellos proyectos más exitosos y analizar y generalizar cómo reparten el tiempo entre las fases del proyecto.

Siguiendo el marco de trabajo que define el estándar (basándonos en PSM), **elabore una propuesta de una lista de atributos, medidas base, medidas derivadas e indicadores a tener en cuenta para plasmar la estrategia planteada.** (2 puntos)

Siguiendo PSM vamos a aplicar una estrategia en dos pasos: **Análisis de proyectos históricos para identificar comportamiento exitoso y determinar indicador, seguimiento en proyectos nuevos del indicador.** Con las métricas de las que disponemos, para poder aplicar la estrategia planteada una de las primeras decisiones será como seleccionar los proyectos más exitosos.

Debemos tener claro que un proyecto exitoso es el que maximiza la eficiencia de los recursos, por lo tanto, de las mediciones registradas, podríamos seleccionar las siguientes:

- M6 Productividad y/o M8 Velocidad de entrega.

Si ordenamos los proyectos registrados por M6 y M8, podemos seleccionar el primer cuartil, el primer 25%. A partir de esto, analizamos el reparto de tiempo, con M1 y M2.

Por tanto, de manera esquemática:



Lista de atributos: Transacciones, número de GLDs, número de DTE, tiempo total, tiempo fase, ...

Ejercicio 3: Métricas Orientadas a Objeto (2 puntos)

Dado el siguiente fichero main.cpp:

```
#include <iostream>

using namespace std;

typedef char String[50]; //Data type for Strings
#define MAX_ACCOUNTS 100 //Number of Account s

class Account //Bank account
{
    float balance;
    int NumAccount;
    bool Blocked; //true if is blocked
public:
    Account();
    Account(int pNo, float pSal);
    bool UpdateBalance(int pSal);
    void UpdateBlock(bool pBlog);
    void UpdateNumAccount(int nAcc);
    float GetBalance();
    int GetNumAccount();
    bool IsBlocked();
};

Account::Account(){
    balance=0;
    NumAccount=0;
    Blocked=false;
}

Account::Account(int pNo, float pSal){
    NumAccount=pNo;
    balance=pSal;
    Blocked=false;
}

bool Account::UpdateBalance(int pSal){
    if(IsBlocked())
        return false;
    else{
        balance=pSal;
        return true;
    }
}

void Account::UpdateNumAccount(int nAcc){
    NumAccount=nAcc;
}

void Account::UpdateBlock(bool pBlog){
    Blocked=pBlog;
}

float Account::GetBalance(){
    return balance;
}

int Account::GetNumAccount(){
    return NumAccount;
}

bool Account::IsBlocked(){
    return Blocked;
}

int SearchAccount(Account Accs[MAX_ACCOUNTS], int
NAccounts, int NumAccount){
    bool found=false;
    int i=0;
    while(!found && i<=NAccounts){
        if(Accs[i].GetNumAccount()==NumAccount)
            found=true;
        else i++;
    }
    if(found) return i;
    else return -1;
}

int MenuAccounts(){
    int choice;
    cout << "\n Account's Management Menu";
    cout << "\n 1 Add an account to a client";
    cout << "\n 2 Show client accounts";
    cout << "\n 3 Delete a client account";
    cout << "\n 4 Modify Balance of an account";
    cout << "\n 5 Modify Account Status";
    cout << "\n 6 Exit";
    cout << "\n Choose Option:";
    cin >> choice;
    return choice;
}
```

```
int main()
{
    Account AccountData[MAX_ACCOUNTS];
    int nAccounts=0;
    int auxNumAccount;
    int choice,pos;
    do{
        choice=MenuAccounts();

        switch(choice){
            case 1: //Add an account
                if(nAccounts<MAX_ACCOUNTS){
                    cout << "\n Please, input the # of
the new account: ";
                    cin >> auxNumAccount;
                    //It exists?
                    if(SearchAccount(AccountData,
nAccounts, auxNumAccount)==-1){
                        cout << "\n Input the balance:
";
                        float auxBalance;
                        cin >> auxBalance;

                        //AccountData[nAccounts+1].Account(auxNumAccount,auxBala
nce);

                        AccountData[nAccounts].UpdateBalance(auxBalance);

                        AccountData[nAccounts].UpdateNumAccount(auxNumAccount);

                        AccountData[nAccounts].UpdateBlock(false);
                        nAccounts++;
                    }else{
                        cout << "\n Account already
exists";
                    }
                }else{
                    cout << "\n maximum number of
accounts reached";
                }
                break;


            case 2: //Show client accounts
                for(int i=0; i<nAccounts; i++){
                    cout << "\n # Account: " <<
AccountData[i].GetNumAccount();
                    cout << "\n Balance: " <<
AccountData[i].GetBalance();
                    cout << "\n Blocked: " <<
AccountData[i].IsBlocked();
                }
                break;

            case 3:
                cout << "\n Please, input the # of
the account to be deleted: ";
                cin >> auxNumAccount;
                pos=SearchAccount(AccountData,
nAccounts, auxNumAccount);
                if(pos!=-1)
                    cout << "\n Error, account to be
deleted doesn't exist";
                else
                    for(int i=pos; i<nAccounts; i++){
                        AccountData[i].UpdateNumAccount(AccountData[i+1].GetNumA
ccount());

                        AccountData[i].UpdateBalance(AccountData[i+1].GetBalance
());

                        AccountData[i].UpdateBlock(AccountData[i+1].IsBlocked())
;
                    }
                nAccounts--;
                break;

            // case 4:
            // case 5:
            // case 6:
            default: cout << "\n Incorrect option. Try
again.";
                break;
        }
    }while(choice!=6);
}
```

| | | |
|---|---|---|
|  | 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 | 6/2/2015 |

Apellidos y nombre: _____ DNI: _____

- a) Calcule la métrica Complejidad Ciclomática del método `main()` (0,5 puntos)

$$V(G)=c+1=9 \text{ (3 case, 3 if, 1 do..while, 2 for)+1=10}$$

- b) Calcule la métrica WMC (Weighted methods per class) de la clase `Account`, ponderando mediante complejidad ciclomática los métodos de manera individual. (0,75 puntos)

$$WMC = \sum_{i=1}^n c_i = 9$$

`Account();` --> complejidad 1
`Account(int pNo, float pSal);` --> complejidad 1
`bool UpdateBalance(int pSal);` --> $c+1 = 1$ if +1 = 2
`void UpdateBlock(bool pBloq);` --> 1
`void UpdateNumAccount(int nAcc);` --> 1
`float GetBalance();` --> 1
`int GetNumAccount();` --> 1
`bool IsBlocked();` --> 1

Las funciones genéricas `SearchAccount` y `MenuAccount`, no pertenecen a la clase y no deben ser tenidas en cuenta.

- c) ¿Qué nos indicaría un número relativamente alto de la métrica WMC? (0,75 puntos)

Complejidad, que esa clase puede ser problemática en cuanto al mantenimiento, con propensión a errores.

Ejercicio 4. Los gráficos en SCRUM (3 puntos).

Nos encontramos en medio de un proyecto que sigue la metodología Scrum. En el siguiente gráfico se recoge el seguimiento del desarrollo seguido.

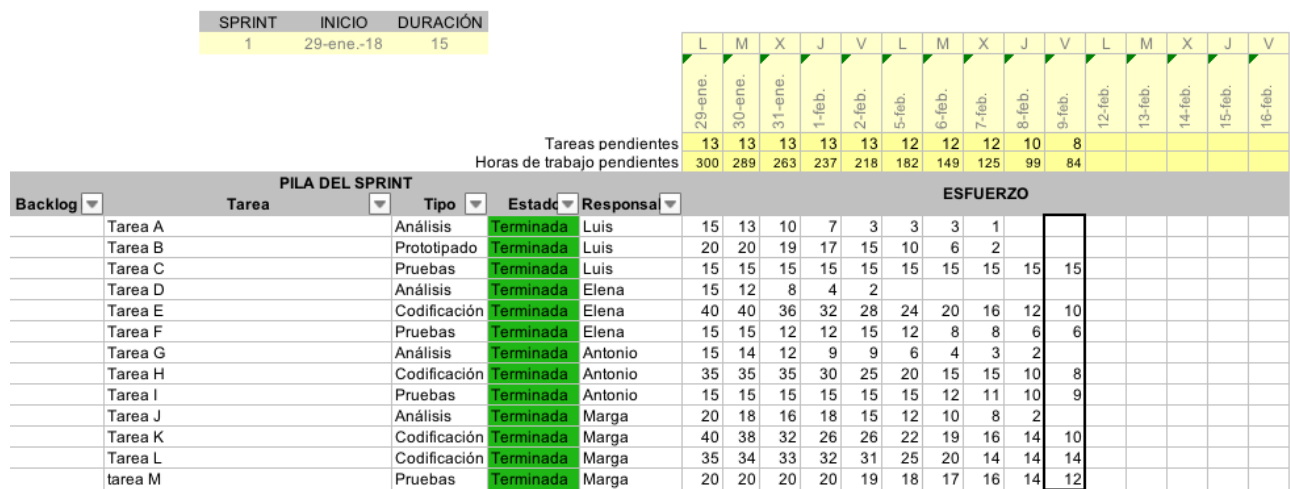
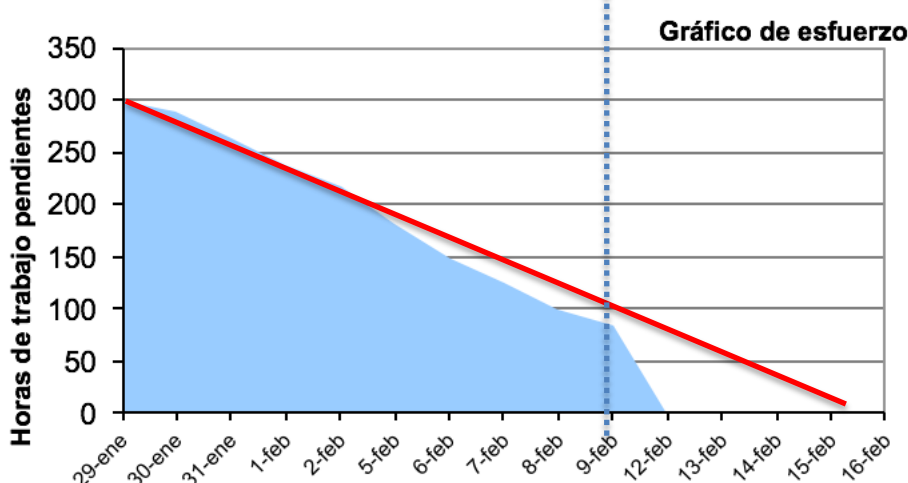


Ilustración 1. Gráfico Burn-down

La iteración está planificada en 3 semanas con una velocidad ideal de 300 puntos. Se pide:

- a) Desarrolle el **gráfico Burn-down** para el esfuerzo pendiente y el esfuerzo ideal.

(1 puntos)



- b) Nos encontramos en el día 9. ¿Qué podemos decir con respecto a la velocidad y la estimación?, ¿qué decisiones tomaría a nivel de dirección de proyectos para el tiempo restante?

(1,5 puntos)

Con respecto a la velocidad y la estimación, podemos decir que al comienzo del proyecto la velocidad está por debajo de la ideal, quizás por subestimación, pero que a partir del día 5-6 ocurre lo contrario, aumenta la velocidad que se sitúa por encima incluso de la ideal.

En el día 9, a la vista del gráfico de seguimiento, observamos que el equipo está desarrollando las funcionalidades previstas a una velocidad mayor de lo esperado, lo que hace suponer que ha habido una sobreestimación del esfuerzo. A nivel de dirección de proyectos, una medida razonable sería adelantar alguna tarea (o historia de usuario) de la siguiente iteración a ésta.

- c) Indique cuál es el empleado más productivo en este sprint y el que menos. Justifique su respuesta.

(0,5 puntos)

Dado que el cuadro de seguimiento del proyecto en proyectos Scrum se basa en el esfuerzo pendiente, para determinar la productividad individual vamos a evaluar la diferencia entre el esfuerzo pendiente al comienzo de la iteración y el esfuerzo pendiente en el día 9, para cada empleado.

| Empleado | Esfuerzo pendiente inicial | Esfuerzo pendiente día 9 | Diferencia |
|----------|----------------------------|--------------------------|------------|
| Luis | 50 | 15 | 35 |
| Elena | 70 | 16 | 54 |
| Antonio | 65 | 17 | 48 |
| Marga | 115 | 36 | 79 |

Marga es la empleada que más ha disminuido su esfuerzo pendiente, luego la de mayor velocidad y por tanto, más productiva. Por el contrario, Luis es el caso contrario.