PSP0.1 Ejercicio 3A

1. Resumen del plan del proyecto nivel PSP0.1

PSP0.1 Project Plan Summary - Program 3A

Student	1-Rubén Ignacio Couoh Ku		Date	26/01/2017
Program	Object Counter		Program#	3A
Instructor	Carlos Mojica		Language	NodeJS v6.9.4

	Plan	Actual	To Date	To Date%
Program Size (LOC)				
Base(B)		73		
Deleted(D)		2		
Modified(M)		5		
Added(A)		55		
Reused(R)		0	0	
Total N&C (N)	65	60	328	
Total LOC(T)		126	394	
Total New Reused		26	26	
Time in Phase (min.)				
Planning	8	9	28	9.0
Design	2	5	10	3.2
Code	69	43	197	63.8
Compile	8	1	19	6.0
Test	11	4	28	8.9
Postmortem	7	14	28	9.1
Total	105	76	309	100.0
Defects Injected				
Planning		0	0	0.0
Design		0	0	0.0
Code		1	12	100.0
Compile		0	0	0.0
Test		0	0	0.0
Total Development		1	12	100.0
Defects Removed				
Planning		0	0	0.0
Design		0	0	0.0
Code		0	0	0.0
Compile		1	10	83.3
Test		0	2	16.7
Total Development		1	12	100.0
After Development		0	0	

2. Resumen del plan del proyecto nivel PSP0.1 2A

PSP0.1 Project Plan Summary - Program 2A

Student	1-Rubén Ignacio Couoh Ku		Date	19/02/2017
Program	LOC Counter		Program#	2A
Instructor	Carlos Mojica		Language	NodeJS v6.9.4

	Plan	Actual	To Date	To Date%
Program Size (LOC)				
Base(B)		0		
Deleted(D)		0		
Modified(M)		0		
Added(A)		83		
Reused(R)		0	0	
Total N&C (N)	80	83	268	
Total LOC(T)		83	268	
Total New Reused		0	0	
Time in Phase (min.)				
Planning	10	9	19	8.0
Design	1	4	5	2.0
Code	97	57	154	66.1
Compile	14	4	18	7.5
Test	9	14	23	10.0
Postmortem	4	11	15	6.3
Total	135	98	233	100.0
Defects Injected				
Planning		0	0	0.0
Design		0	0	0.0
Code		4	11	100.0
Compile		0	0	0.0
Test		0	0	0.0
Total Development		4	11	100.0
Defects Removed				
Planning		0	0	0.0
Design		0	0	0.0
Code		0	0	0.0
Compile		3	9	81.8
Test		1	2	18.2
Total Development		4	11	100.0
After Development		0	0	

3. Forma de registros PIPs.

Propuesta de Mejora de Proceso (PIP)

Nombre: Rubén Ignacio Couoh Ku Fecha: 26/10/2017
Proyecto: 3A
Proceso: PSP0.1 Elementos: Script del proceso - desarrollo

Número de PIP	**Descripción del Problema:**
1	En el script de desarrollo en la sección de diseño le falta ordenar los pasos a seguir, desde mi punto de vista quedaría más claro, me causa confusión.

PROPUESTA

# PIP	**Descripción de la Propuesta**
1	Cambiar el orden de los pasos a ejecutar en la sección de diseño en el script de desarrollo PSP0.1. Cambiar el orden de los siguientes pasos: • Implementar el diseño **siguiendo el Estándar de Codificación**. • Revisar los requerimientos y producir un diseño que los satisfaga. Por : • Revisar los requerimientos y producir un diseño que los satisfaga. • Implementar el diseño **siguiendo el Estándar de Codificación**.

4. Forma de registro de tiempos.

Project	Phase	Date	Start	Int.	Stop	Delta	Comments
3	PLAN	01/26/17	13:25:25		13:34:34	9.1	Validación de requerimientos y estimación de tamaño y tiempo.
3	DLD	01/26/17	13:40:28		13:45:38	5.2	Diseño del programa Object Counter; división de tareas: *Contador de líneas de código, *Contador métodos del objeto, *Entrada de datos, *Salida de resultados en pantalla, *Integración de todo lo anterior.
3	CODE	01/26/17	13:59:58		14:01:35	1.6	Reutilización de la clase LinesOfCodeCounter.
3	CODE	01/26/17	14:09:04		14:29:06	20.0	Codificación del contador de métodos de objeto.
3	CODE	01/26/17	14:31:04		14:32:13	1.2	Se utilizó el código de entrada de archivos del programa 2A.
3	CODE	01/26/17	14:38:32		14:43:27	4.9	Salida en pantalla del análisis de los resultados.
3	CODE	01/26/17	14:48:28		15:03:57	15.5	Integración de *Contador de líneas de código, *Contador métodos del objeto, *Entrada de datos, *Salida de resultados en pantalla.
3	COMPILE	01/26/17	15:06:59		15:08:01	1.0	Clase ObjectMethodsCounter -> Simbolo inesperado en el operador ternario, se esperaba ":" y se coloco "|": numberOfMethods = found ? found.length : 0;
3	TEST	01/26/17	16:15:47		16:17:22	1.6	Se probó el programa con el código fuente del programa 1A, se verifico la salida del programa 3A.
3	TEST	01/26/17	16:18:53		16:19:59	1.1	Se probó el programa con el código fuente del programa 2A, se verifico la salida del programa 3A.
3	TEST	01/26/17	16:22:24		16:23:59	1.6	Se probó el programa con el código fuente del programa 3A, se verifico la salida del programa 3A.
3	PM	01/26/17	16:39:22		16:50:49	11.5	Verificando líneas de codigo en las formas de resumen..., interrupción duda sobre trasmisión de video en tiempo real (Juan)
3	PM	01/26/17	17:09:28		17:11:36	2.1	Continuación de la validación de líneas de código en las formas de resumen.

5. Forma de registro de defectos.

Project	Date	Num	Type	Injected	Removed	FixTime	Fix Ref.	Description
3	26/01/2017	12	20	CODE	COMPILE	1.0		Se corrigió la sintaxis del operador ternario que se encuentra en el método count de la clase ObjectMethodsCounter.

6. Código fuente del programa.

```
/*************************************************************************/
/*  Name:        Rubén Ignacio Couoh Ku.                                 */
/*  Date:        26/01/2017                                              */
/*  Description: Contador de líneas de código (LOC)                      */
/*************************************************************************/

const FS       = require('fs');
const READLINE = require('readline');
const PATH     = require('path');

/*************************************************************/
/*  pattern filtra todos los comentarios cortos y largos    */
/*      Comentario corto: //                                 */
/*      Comentario largo: \/*... *\/                         */
/*************************************************************/
let pattern = /^\s*\/{2,}|^\s*\/\*.*\*\/\s*$|^\s*.{0}\s*$/;

class LinesOfCodeCounter {

    /*************************************************************************************************/
    /*  Reuse Instructions                                                                           */
    /*      count(file, cb)                                                                          */
    /*      Purpose:     Cuenta las líneas de código del ${file}                                     */
    /*      Limitations: NA                                                                          */
    /*      Return:      Regresa un JSON {name, linesOfCode} en la funcion callback.                 */
    /*************************************************************************************************/

    static count(file, cb)
    {
        let name = PATH.basename(file, '.js');
        let linesOfCode = 0;
        let summary = {};
        let rl = READLINE.createInterface({
            input: FS.createReadStream(file, 'utf-8')
        });

        rl.on('line', (line) => {
            //  Si no es comentario o línea en blanco la cuenta como línea de código.
            if (!pattern.test(line)) {
                linesOfCode++;
            }
        });

        rl.on('close', () => {
            summary = {name, linesOfCode}
            cb(null, summary);
        });
    }
}

module.exports = LinesOfCodeCounter;
```

```
/*****************************************************************************/
/*  Name:        Rubén Ignacio Couoh Ku.                                     */
/*  Date:        26/01/2017                                                  */
/*  Description: Contador de métodos de objeto.                              */
/*****************************************************************************/

const FS   = require('fs');
const PATH = require('path');


/*************************************************************/
/*  pattern filtra las firmas de los metodos                 */
/*      example: static main(arg1, arg2)                     */
/*      example: count(arg1, arg2)                           */
/*************************************************************/

let pattern = /\s*[A-Za-z]*\s*(_?[A-Za-z0-9]+\s*\(\s*([A-Za-z]+\,\s*)*\s*[A-Za-z]*\s*\))(?=\s*\r\n\s*{\s*)/g;

class ObjectMethodsCounter {

    /*************************************************************************************************************/
    /*  Reuse Instructions                                                                                       */
    /*      count(file, cb)                                                                                      */
    /*      Purpose:     Cuenta los métodos de una clases.                                                       */
    /*      Limitations: NA                                                                                      */
    /*      Return:      Regresa un JSON {name, numberOfMethods} en la función callback.                         */
    /*************************************************************************************************************/

    static count(file, cb)
    {
        FS.readFile(file, 'utf-8', (err, content) => {
            let name = '';
            let numberOfMethods = 0;
            let summary = {};
            if (err) {

                console.error(err);
            } else {
                let matches = content.match(pattern);

                // Descarta todas las funciones.
                let found   = matches.filter((match) => {

                    return !match.includes('function');
                });

                numberOfMethods = found ? found.length : 0;
            }

            name = PATH.basename(file, '.js');
            summary = {name, numberOfMethods};
            cb(null, summary);
        });
    }
}

module.exports = ObjectMethodsCounter;
```

```
/*****************************************************************************************/
/*  Name:        Rubén Ignacio Couoh Ku.                                                 */
/*  Date:        26/01/2017                                                              */
/*  Description: Programa utilizado para contar líneas de código y número de metodos.    */
/*****************************************************************************************/

const GLOB           = require('glob');
const FS             = require('fs');
const PATH           = require('path');

const LINESCOUNTER   = require('./LinesOfCodeCounter.js');
const METHODSCOUNTER = require('./ObjectMethodsCounter.js');

class Main {

    static _getFilesByExtension(folder, extensions, cb)
    {
        let files = '*.+({extensions})'.replace('{extensions}', extensions.join('|'));
        let pattern = '{folder}/{files}'
                        .replace('{folder}', folder)
                        .replace('{files}', files);

        GLOB(pattern, cb);
    }

    static _printSummary(header, summaries)
    {
        let linesOfCode = 0;
        console.log();
        console.log('*****************************************************************************');
        console.log(`*\tPrograma: ${header}`);

        console.log('*****************************************************************************');

        summaries.forEach((summary) => {
            linesOfCode += summary.linesOfCode;
            console.log(`*\t${summary.name}\t# Methods ${summary.numberOfMethods}\t# LOC ${summary.linesOfCode}`);
            console.log('*****************************************************************************');
        });

        console.log(`*\t\t\tTotal LOC:\t${linesOfCode}`);
        console.log('*****************************************************************************');
    }
```

```
44
45      static main()
46      {
47          //  Carpeta donde se encuentra el programa
48          //let folder = './../../PSP0/1A';
49          //let folder = './../../PSP0.1/2A';
50          let folder = './../../PSP0.1/3A';
51          let header = PATH.basename(folder);
52          // Tipos de archivos en los cuales se desean contar las líneas de código.
53          // Pueden existir archivos e configuración que no se desean contar.
54          let extensions = ['js'];
55
56          Main._getFilesByExtension(folder, extensions, (err, files) => {
57              if (err) {
58                  console.error(err);
59                  return;
60              }
61              countLineOfCodeAndNumberMethodsByFile(files, processSumaries);
62
63          });
64
65          function countLineOfCodeAndNumberMethodsByFile(files, cb)
66          {
67              let remaining = files.length;
68              let summaries = [];
69
70              if (files.length) {
```

```
70              if (files.length) {
71
72                  files.forEach(function (file) {
73
74                      LINESCOUNTER.count(file, (err, linesSummary) => {
75
76                          if (err) {
77                              console.log(err);
78                          }
79
80                          METHODSCOUNTER.count(file, (err, methodsSummary) => {
81
82                              if (err) {
83                                  console.log(err);
84                              }
85                              let summary = {};
86
87                              Object.assign(summary, linesSummary, methodsSummary);
88                              summaries.push(summary);
89
90                              if (--remaining === 0) {
91                                  cb(summaries);
92                              }
93                          });
94                      });
95                  });
96              } else {
97
98                  cb(summaries);
99              }
100         }
101
102         function processSumaries(summaries)
103         {
104             Main._printSummary(header, summaries);
105         }
106     }
107 }
108
109 Main.main();
```

7. Reporte R3

Defect Densities				Compile and Test Defects			
Program Number	New and Changed LOC	Total Defects	Defects per KLOC	Defects found in compile	Compile defects per KLOC	Defects found in test	Test defects per KLOC
1	185	7	38	6	32	1	5
2	83	4	48	3	36	1	12
3	60	1	17	1	17	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
Totals	328	12	37	10	30	2	6

Defect Fix Times				
		Defects found in compiling	Defects found in testing	Total defects found
Defects injected in designing	Tot. fix time	0	0	0
	Tot. defects	0	0	0
	Avg. fix time	0	0	0
Defects injected in coding	Tot. fix time	19	9	28
	Tot. defects	10	2	12
	Avg. fix time	2	5	2
Total defects injected	Tot. fix time	19	9	28
	Tot. defects	10	2	12
	Avg. fix time	2	5	2

8. Interfaz de usuario.

```
**************************************************************************
*       Programa: 3A
**************************************************************************
*       Main    # Methods 3     # LOC 75
**************************************************************************
*       LinesOfCodeCounter      # Methods 1     # LOC 25
**************************************************************************
*       ObjectMethodsCounter    # Methods 1     # LOC 26
**************************************************************************
*                       Total LOC:      126
**************************************************************************
```

9. Resultados.

Entrada: Programa 1A

Salida:

```
**************************************************************
*       Programa: 1A
**************************************************************
*       Main    # Methods 1     # LOC 71
**************************************************************
*       Node    # Methods 1     # LOC 8
**************************************************************
*       RMath   # Methods 3     # LOC 36
**************************************************************
*       List    # Methods 8     # LOC 70
**************************************************************
*                       Total LOC:      185
**************************************************************
```

Entrada: Programa 2A

Salida:

```
*************************************************************************
*        Programa: 2A
*************************************************************************
*        LinesOfCodeCounter       # Methods 1      # LOC 23
*************************************************************************
*        Main     # Methods 3      # LOC 60
*************************************************************************
*                          Total LOC:       83
*************************************************************************
```

Entrada: Programa 3A

Salida:

```
*************************************************************************
*        Programa: 3A
*************************************************************************
*        Main     # Methods 3      # LOC 75
*************************************************************************
*        LinesOfCodeCounter       # Methods 1      # LOC 25
*************************************************************************
*        ObjectMethodsCounter     # Methods 1      # LOC 26
*************************************************************************
*                         Total LOC:       126
*************************************************************************
```

10.Tabla de resultados:

# de programa	Nombre del Objeto	Número de métodos	LOC del Objeto	LOC totales del programa
1A	Node	1	8	
	List	8	70	
	RMath	3	36	
	Main	1	71	
				185
2A	LinesOfCodeCounter	1	23	
	Main	3	60	
				83
2A	LinesOfCodeCounter	1	25	
	ObjectMethodsCounter	1	26	
	Main	3	75	
				126