UNIVERSIDAD DE LAS FUERZAS ARMADAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
INGENIERÍA DE SOFTWARE

Desarrollo de Software Aplicado a la Interculturalidad

Buenas prácticas en la documentación y trazabilidad de requisitos en el DSG

NRC: 23398

Integrantes:
Christopher Bazurto
Stephen Drouet
Leonardo de la Cadena
Ricardo Rivadeneira
Andy Pilozo

FECHA DE ENTREGA: 09/06/2025

Índice

1. Introducción

En proyectos de Desarrollo de Software Global (DSG), donde equipos dispersos geográficamente trabajan sobre un mismo producto, la documentación y la trazabilidad de requisitos se convierten en pilares fundamentales. Una buena documentación asegura que todos los participantes conozcan qué se va a construir y por qué; la trazabilidad permite rastrear cada funcionalidad desde su origen hasta su implementación y validación, reduciendo riesgos de desviaciones, malentendidos o pérdidas de alcance.

2. Desarrollo

2.1. Factores de estimación

Los factores de estimación son variables que influyen directamente sobre el coste, esfuerzo y duración de un proyecto. En entornos globales, destacan:

- Tamaño del software (Líneas de Código, Funciones).
- Complejidad de los requisitos.
- Experiencia del equipo y familiaridad con la tecnología.
- Madurez de procesos y herramientas de colaboración.
- Heterogeneidad cultural y horaria.

Cada uno de estos factores puede aumentar o disminuir la previsión inicial de recursos y tiempo, por lo que deben identificarse y cuantificarse desde la planificación.

2.2. Modelos de estimación

COCOMO (Constructive Cost Model) es un modelo algorítmico de estimación de costos y esfuerzo en proyectos de software, desarrollado por Barry W. Boehm en 1981. Está basado en una regresión empírica usando datos de proyectos reales, y permite predecir tres dimensiones clave: esfuerzo, tiempo y tamaño del equipo necesarios para completar un proyecto.

- **El esfuerzo (E):** se mide en persona-meses (PM).
- **Duración (TDEV):** se mide en meses.
- **Personal (P):** se obtiene dividiendo esfuerzo entre duración.

Niveles y modos del modelo

COCOMO se presenta en tres niveles de detalle:

1. **Básico**: sólo utiliza KLOC y el modo de proyecto (orgánico, semidetach, empotrado).

2. **Intermedio**: agrega entre 15 y más multiplicadores de esfuerzo ("cost drivers") para ajustar la estimación por factores como experiencia del equipo, calidad requerida, entorno, etc.
3. **Detallado (Advanced)**: considera también características por fase o módulo

Los modos definen distintos tipos de proyecto:

- **Orgánico**: proyectos pequeños, equipo experimentado, entorno familiar.
- **Semidetach**: complejidad y equipo mixto.
- **Empotrado (embedded)**: sistemas con fuertes restricciones técnicas.

Fórmulas para modo orgánico en el modelo básico

Para proyectos orgánicos:

$$Esfuerzo\ (E)\ =\ a\ *\ (KLOC)^{b}$$

$$Duración\ (TDEV)\ =\ c\ *\ E^{d}$$

$$Personal\ (P)\ =\ \frac{E}{TDEV}$$

$$a\ =\ 2.4,\ b\ =\ 1.05, c\ =\ 2.5, d\ =\ 0.38$$

¿Para qué sirve COCOMO?

1. **Planificación preliminar**: ofrece estimaciones tempranas para presupuesto, calendario y recursos.
2. **Comparativas históricas**: su base empírica permite evaluar escenarios futuros similares
3. **Ajuste por factores**: las versiones intermedio/detallado permiten adaptar la estimación al contexto específico (tecnología, experiencia, requisitos de calidad...).
4. **Mejora continua**: la familia COCOMO II permite calibrar el modelo según datos reales, mejorando la precisión

2.3. Herramientas de estimación

Las herramientas de estimación para proyectos distribuidos se agrupan en suites paramétricas que aplican modelos como COCOMO II, SEER-SEM o SLIM-Estimate; contadores de tamaño funcional y métricas estáticas por ejemplo ScopeMaster, CAST Highlight o utilidades de conteo de líneas de código como cloc que cuantifican Puntos

Función o KLOC desde requisitos y código; y plataformas colaborativas integradas en sistemas de gestión ágil (Jira, Azure DevOps, Planning Poker Online) que facilitan estimaciones por consenso con registro de la velocidad histórica del equipo y soporte multi zona horaria.

Comandos Cloc:

cloc . --exclude-list-file=.clocignore --include-lang=Dart,JavaScript

```
github.com/AlDanial/cloc v 2.04  T=23.46 s (110.1 files/s, 23931.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                    2547          39549         153568         364815
Dart                            36            376            291           2928
-------------------------------------------------------------------------------
SUM:                          2583          39925         153859         367743
-------------------------------------------------------------------------------

C:\Users\USUARIO\Documents\SEMESTRE\INTERCULTURALIDAD\transporte>
```

cloc . --exclude-list-file=.clocignore

```
github.com/AlDanial/cloc v 2.04  T=41.08 s (138.4 files/s, 26757.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                    2547          39549         153568         364815
TypeScript                    1329          23632         155430         123513
JSON                           596            147              0          70726
Markdown                       476          24076            505          56586
XML                            201           2664            233          13487
Text                            94            429              0           9601
Protocol Buffers               210           4651          21921           9469
C/C++ Header                    13           1379            968           7626
Dart                            36            376            291           2928
C++                             12            400            243           2064
HTML                             5            249             12           2054
CMake                           25            278            151           1038
YAML                            43             74            128            777
C                                1            131             62            610
DOS Batch                       19             50              2            341
Bourne Shell                    17             57             27            308
PowerShell                      13             13             65            286
Python                           5              0              4            138
GLSL                             1              0              0            126
Bourne Again Shell               1             19             20            121
Java                             4              6             20             90
Gradle                           3             16             10             77
INI                              4             20              0             74
Swift                            6             15              7             71
Windows Resource File            1             23             29             69
Objective-C                      1             15              4             65
make                             4             26              4             54
Properties                      11              0              5             42
D                                5              0              0              5
SQL                              2              0              0              4
Kotlin                           1              2              0              3
-------------------------------------------------------------------------------
SUM:                          5686          98297         333709         667168
-------------------------------------------------------------------------------
```

JavaScript constituye el núcleo del proyecto: más de la mitad de las líneas provienen de servicios, scripts y componentes de interfaz escritos en este lenguaje. Le sigue TypeScript con casi una quinta parte del total, lo que delata una estrategia de migración gradual o la convivencia de ambos lenguajes para aprovechar tipado estático sin abandonar código legado.

Una porción relevante de los conteos corresponde a archivos JSON y XML, utilizados para configuración, mocks o intercambio de datos. Aunque estas líneas no aportan lógica ejecutable, inflan el tamaño aparente del repositorio y pueden distorsionar métricas si se toman sin filtrar.

3. Aplicación práctica

Primero se procede a identificar los requisitos

ID RF	Tema	Como un..	Necesito

REQ001	Registrar cliente	Cliente	Poder registrarme en la aplicación para poder utilizarla
REQ002	Iniciar sesión	Cliente	Utilizar las funcionalidades de la aplicación
REQ003	Servicio de transporte	Cliente	Trasladarse de un lado a otro
REQ004	Servicio de transporte	Cliente	Cancelar la solicitud del servicio
REQ005	Servicio de transporte	Cliente	Ver el listado de conductores disponibles
REQ006	Servicio de transporte	Conductor	Ver que cliente ha solicitado un servicio de transporte
REQ007	Geolocalización	Conductor	Ver la ubicación del cliente en el mapa
REQ008	Geolocalización	Cliente	Ver la ubicación en tiempo del conductor en el mapa
REQ009	Servicio de transporte	Conductor	Finalizar la carrera
REQ010	Registro de conductor	Conductor	Registrarse en la aplicación
REQ011	gestion de cooperativas	Administrador	Agregar cooperativas
REQ012	gestion de cooperativas	Administrador	Gestionar de usuarios
REQ013	Codigo de seguridad	Conductor	Recibir código de cliente
REQ014	Funcionalidad intercultural	Usuarios en general	Poner una frase de respeto hacia las diversas culturas al iniciar la aplicación

LOC por PF de cada lenguaje

Lenguaje de programación	**LOC por PF (Aproximado)**
Javascript	47
Flutter/dart	31

Asignación de complejidad y cálculo del KLOC

Se define para cada requisito un valor de complejidad entre 1 (muy bajo) y 10 (muy alto), y se asume que el LOC Javascript + LOC Dart = 47+31 = 78

Tipo de Componente	**Descripción**	**Cantidad**	**Complejidad**	**PF Unidad**	**Total PF**
Entradas Externas (EE)	Registro cliente, login, solicitar/cancelar servicio, registrar conductor, agregar cooperativas, gestionar usuarios	6	Media	4	24
Salidas Externas (SE)	Código de seguridad al conductor, mensaje intercultural	2	Baja	4	8
Consultas Externas (CE)	Ver conductores disponibles, ver solicitudes por conductor, ver ubicación de cliente/conductor	3	Media	4	12

Archivos Lógicos Internos (ALI)	Base de datos: usuarios, conductores, solicitudes, cooperativas, ubicaciones	4	Alta	10	40
Interfaces Externas (IE)	Uso de APIs para mapas/geolocalización (Google Maps, etc.)	1	Media	7	7

$$Total\ de\ PF\ =\ 91$$

$$KLOC = \frac{91*78}{1000} = 7.098 KLOC$$

Esfuerzo (E): Fórmula que calcula cuántos meses de trabajo de una sola persona tomaría desarrollar el software. No se refiere a la duración, sino a la cantidad de esfuerzo humano total requerido

$$E = a * (KLOC)^b$$

$$E = 2.4 * (7.098)^{1.05}$$

$$E = 18.78901$$

Como el grupo de trabajo es de 5 personas, el Esfuerzo estimado por persona es de 3.757802 meses

Tiempo de desarrollo (T): Calcula la duración del proyecto en meses reales, considerando que el esfuerzo no se realiza por una sola persona, sino por un equipo.

$$T = c * E^d$$

$$T = 2.5 * (18.78901)^{0.38}$$

$$T = 7.62120$$

Como el equipo es de 5 personas dividimos este valor para 5, dando un resultado de: 1.52424

Siendo el tiempo de realización de proyecto 1 mes y medio

Personal necesario (P): Nos dice cuántas personas deberían trabajar en el proyecto para completarlo en el tiempo estimado.

$$P = E/T$$

$$P = 18.78901/7.62120$$

$$P = 2.46536$$

Productividad (PROD): Mide cuán eficiente es el equipo de desarrollo, es decir, cuántas líneas de código produce cada persona en promedio por mes.

$$PROD = KLOC/persona - mes$$

$$PROD = 7.098/18.78901$$

$$PROD = 0.37777\ KLOC/persona - mes$$

Costo total (CT) : Es la estimación del costo en dinero, considerando el salario promedio mensual de cada desarrollador. (Considerando que el promedio de sueldo mensual para cada programador es de $600 USD)

$$CT = E * salario\ promedio\ mensual$$

$$CT = 18.78901 * 250$$

$$CT = 4697.25$$

El costo total del proyecto es : $ 4697.25 USD

4. Conclusiones

- Las estimaciones COCOMO para un sistema de 7,1 KLOC (≈ 91 PF) sitúan el esfuerzo en 18,8 persona/meses, la duración en 7,6 meses, el tamaño del equipo en 2,5 desarrolladores y el coste en unos 11,3 mil USD.
- El tamaño del software, la complejidad de los requisitos, la experiencia del equipo, la madurez de procesos y la diversidad cultural y horaria son los factores que más repercuten en esfuerzo, coste y calendario en entornos de desarrollo global.
- Los niveles básico, intermedio y detallado de COCOMO, junto con sus modos orgánico, semi detached y empotrado, muestran cómo el modelo refina sus previsiones al incorporar multiplicadores y restricciones específicas.

5. Referencias

- Damian, D., & Moitra, D. (2006). Global software development: How far have we come? IEEE Software, 23(5), 17–19. https://doi.org/10.1109/MS.2006.126
- Gotel, O. C. Z., & Finkelstein, A. C. W. (1994, abril). An analysis of the requirements traceability problem. En Proceedings of the First International Conference on Requirements Engineering (pp. 94–101). IEEE. https://doi.org/10.1109/ICRE.1994.292401
- International Function Point Users Group. (2023). Function point counting practices manual (Release 5.1.1). IFPUG.
- Boehm, B. W. (1981). *Software engineering economics*. Prentice-Hall. https://www.geeksforgeeks.org/software-engineering-cocomo-model/