Grupo 1 - Alvarez, Ayrton. Bruschi, Tomás. Gonzalez, Iván. Gonzalez, Joaquín. Plaquin, Leandro.

Calidad de Software

Ingeniería de Software 3 - Actividad 2 - 12/05/2025

TEMARIO

Modularidad de Código
Documentación
Gestión de Logs
Manejo de Excepciones
Pruebas Automatizadas
Desacoplamiento entre Componentes
Estandarización del Código
Control de Versiones y Trazabilidad de Cambios

MODULARIDAD DE CÓDIGO

Tomamos en cuenta **3 categorías** en las que **podemos clasificar a los Módulos del sistema a evaluar** y a cada categoría **le asignamos un puntaje:**

- *Altamente Modularizado*: 1.0 Puntos.
- *Medianamente Modularizado*: 0.7 Puntos.
- *No modularizado*: 0.4 Puntos.

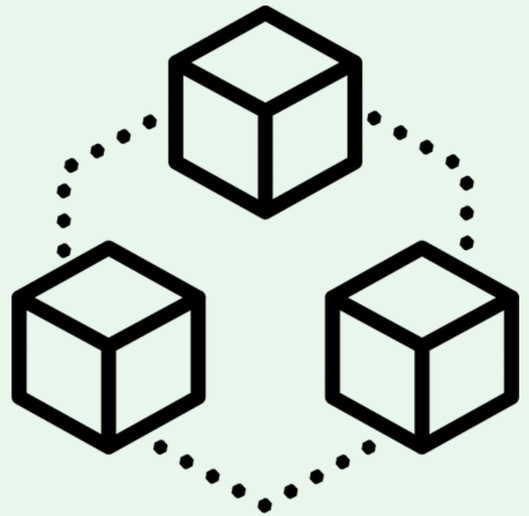
A partir de esto usamos la siguiente **fórmula para medir la Modularidad de Código:**

$$\frac{\text{Cantidad de puntos totales}}{\text{Cantidad de Modulos identificados}}$$

MODULARIDAD DE CÓDIGO

A partir del resultado obtenido, podemos **establecer valores para la escala de la métrica:**

- Satisfactorio
  - *Excede los Requerimientos*: Mayor o igual a 0.90
  - *Rango Aceptado*: Mayor o igual a 0.85
  - *Mínimamente Aceptable*: Mayor o igual a 0.75
- Insatisfactorio
  - *Inaceptable:* Menor a 0.75



DOCUMENTACIÓN

Para analizar la documentación del producto,
se va a tener en cuenta los siguientes criterios:

- **Completitud:**
  - ¿La documentación cubre todos los aspectos necesarios del sistema?
- **Exactitud**:
  - ¿La información reflejada es correcta y está alineada con el software?
- **Claridad**:
  - ¿El contenido es comprensible para su audiencia objetivo?

DOCUMENTACIÓN

Valor por categoría:

- **Completitud:**
  - Funcionalidades documentadas / Funcionalidades totales.
- **Exactitud:**
  - Funcionalidades documentadas correctas / Funcionalidades documentadas.
- **Claridad:**
  - 0,2: Pobre claridad.
  - 0,4: Baja claridad.
  - 0,6: Mediana claridad.
  - 0,8: Buena claridad.
  - 1: Alta claridad.

La calidad final se calcula con la siguiente fórmula:

$$(Completitud \cdot 0.4) + (Exactitud \cdot 0.4) + (Claridad \cdot 0.2)$$

DOCUMENTACIÓN

A partir del puntaje final, la calidad se mide de la siguiente forma:

- **Satisfactorio**
  - **Excede los requerimientos:** Mayor o igual a 0,90
  - **Rango aceptado:** Mayor o igual a 0,85
  - **Mínimamente Aceptable:** Mayor o igual a 0.75
- **Insatisfactorio**
  - **Inaceptable:** Menor a 0.75

GESTIÓN DE LOGS

Objetivo: asegurar trazabilidad, diagnóstico eficiente y soporte a la observabilidad del sistema.

Aspectos clave de calidad:

Información Detallada y Contextual

- Timestamp
- Nivel de Severidad (DEBUG, INFO, ERROR...)
- Módulo/Componente
- Evento específico y resultado
- Usuario
- Host o instancia

Formato Estructurado y Consistente
formato JSON para análisis automatizado.

Correcto Uso de Niveles de Log
Permite filtrar adecuadamente por entorno (producción/desarrollo).

Política de Logs

- Almacenamiento y Retención

GESTIÓN DE LOGS

Metodología de evaluación propuesta:

- **Timestamp (marca temporal del evento):**
  - **1.0** si es preciso y completo
  - **0.5** si le faltan detalles
  - **0.25** si es impreciso o incompleto

- **Formato estructurado:**
  - **1.0** si está definido y siempre se sigue
  - **0.5** si está definido pero no se respeta siempre
  - **0.25** si no hay formato definido

- **Niveles de severidad adecuados:**
  - **1.0** si están bien definidos y son específicos
  - **0.5** si faltan niveles o son poco claros
  - **0.25** si casi no hay clasificación

- **Información esencial sobre el evento:**
  - **1.0** si incluye detalles suficientes para diagnóstico
  - **0.5** si tiene pocos datos útiles
  - **0.25** si apenas aporta información

- **Identificación del componente que genera el log:**
  - **1.0** si se indica de forma clara
  - **0.5** si se menciona de forma ambigua
  - **0.25** si no se identifica el componente

- **Configuración de almacenamiento del log:**
  - **1.0** si está definida explícitamente
  - **0.5** si se usa una configuración por defecto
  - **0.25** si no se configura

Rangos de calidad:

- **Satisfactorio:** 0.8 – 1.0
- **Aceptable:** 0.4 – 0.8
- **Insatisfactorio:** 0.0 – 0.4

$$\frac{\text{Cantidad de puntos totales}}{\text{Cantidad de aspectos del log analizados}}$$

MANEJO DE EXCEPCIONES

Un **producto de software** debe **capturar y manejar correctamente** las **excepciones** sin comprometer su **estabilidad**, asegurando que todos los errores estén **contemplados** y se **traduzcan en respuestas controladas al usuario.**

Basándonos en el **proceso de evaluación** de la norma **ISO/IEC 25040** y las **características de calidad** del modelo **ISO/IEC 25010** construimos una **tabla** que nos permite **medir el Manejo de Excepciones.**

MANEJO DE EXCEPCIONES

Criterio Evaluado	Métrica	Inaceptable	Aceptable	Rango Objetivo	Excede Expectativas
Captura de errores en bloques críticos	% de funciones críticas con try/catch	< 40%	40–69%	70–89%	≥ 90%
Registro estructurado en logs	% de excepciones logueadas con contexto	< 50%	50–69%	70–85%	≥ 86%
Estabilidad del sistema ante errores	N° de caídas por errores no controlados	> 5 en pruebas	3–5 en pruebas	1–2 en pruebas	0
Mensajes de error amigables al usuario	% de errores que devuelven mensaje claro y útil	< 50%	50–74%	75–89%	≥ 90%

PRUEBAS AUTOMATIZADAS

- **Objetivo:**
  - Implementar pruebas automatizadas que aseguren calidad, reduzcan errores y mantengan el desarrollo ágil.
- **Contexto Normativo:**
  - Basado en ISO/IEC 25010 y 25040
  - Evalúa adecuación funcional y mantenibilidad
- **Tipos de pruebas requeridas:**
  - Unitarias: Validan componentes individuales
  - Integración: Verifican interacción entre componentes
  - Funcionales (E2E): Validan flujos completos

PRUEBAS AUTOMATIZADAS

Criterio Evaluado	Métrica	Inaceptable	Aceptable	Rango Objetivo	Excede Expectativas
Cobertura de código (unitarias)	% de líneas de código cubiertas por pruebas unitarias	< 30%	30–59%	60–85%	> 85%
Cobertura de flujos críticos (funcionales)	% de funcionalidades principales cubiertas por pruebas E2E	< 50%	50–74%	75–89%	≥ 90%
Integración en CI/CD	¿Se ejecutan automáticamente en cada commit o merge?	No	Parcial (manual)	Sí, con fallos ocasionales	Sí, estable y con reportes
Detección de regresiones	N° de regresiones no detectadas antes del despliegue	> 5	3–5	1–2	0
Reportes y trazabilidad de resultados	¿Se generan reportes automatizados con resultados de pruebas?	No	Sí, pero incompletos	Sí, con métricas básicas	Sí, detallados y vinculados a tickets/issues

DESACOPLAMIENTO ENTRE COMPONENTES

Para medir el desacoplamiento entre los componentes, hay que tener en cuenta los siguientes criterios para cada componente:

- **Fan-in:** Cuántos componentes dependen de este componente.
- **Fan-out:** Cuántos componentes este componente depende.

El valor de cada categoría es:

- **Fan-in**: $\frac{\text{Componentes dependientes del componente}}{\text{Componentes totales}}$
- **Fan-out**: $\frac{\text{Componentes que el componente depende}}{\text{Componentes totales}}$

DESACOPLAMIENTO ENTRE COMPONENTES

El nivel de desacoplamiento de un componente es:

$$1 - \frac{(\text{Fan in} + \text{Fan out})}{2}$$

El nivel de desacoplamiento del producto es:

$$\frac{\text{Niveles de desacoplamiento de los componentes}}{\text{Cantidad de componentes totales}}$$

A partir del resultado, la calidad se mide:

- **Satisfactorio**
  - **Excede los requerimientos:** Mayor o igual a 0,90
  - **Rango aceptado:** Mayor o igual a 0,85
  - **Mínimamente Aceptable:** Mayor o igual a 0.75
- **Insatisfactorio**
  - **Inaceptable:** Menor a 0.75

ESTANDARIZACIÓN DEL CÓDIGO

Analizando las **líneas de código entregables** podemos reconocer los **lenguajes de programación** utilizados y de esta forma asociar las **guías de estilo y estandarización** que la comunidad tiene en cuenta.

Dependiendo el/los lenguaje/es, **analizaremos el código** con el **Linter** (analizador estático) correspondiente en pos de **encontrar violaciones a los estándares.**

ESTANDARIZACIÓN DEL CÓDIGO

Una vez analizado el código podemos **medir la estandarización usado la siguiente fórmula:**

$$1 - \frac{\text{Lineas con violaciones}}{\text{Total de lineas de codigo entregables}}$$

A partir del resultado obtenido, podemos **establecer valores para la escala de la métrica:**

- Satisfactorio
  - *Excede los Requerimientos*: Mayor o igual a 0.90
  - *Rango Aceptado*: Mayor o igual a 0.80
  - *Mínimamente Aceptable*: Mayor o igual a 0.70
- Insatisfactorio
  - *Inaceptable*: Menor a 0.70

CONTROL DE VERSIONES Y TRAZABILIDAD DE CAMBIOS

- **Todo cambio debe registrarse en el sistema de control de versiones.**

- **Cada commit debe:**
  - Ser claro (describir qué se hizo).
  - Tener referencia a una tarea o issue.
  - Tener una frecuencia adecuada.
  - Tener un tamaño razonable

Calidad	Puntos	Descripción breve
Aceptable	1.0	Claro, con referencia, tamaño y frecuencia ok
Intermedio	0.5	Claro, pero con fallas menores
Inadecuado	0.25	Poco claro, sin referencia, muy grande/disperso

CONTROL DE VERSIONES Y TRAZABILIDAD DE CAMBIOS

$$\frac{\text{Puntaje de todos los commits}}{\text{Cantidad total de commits}}$$

Proyecto	Excede	Aceptado	Mínimo	Inaceptable
Chico	≥ 0.95	≥ 0.80	≥ 0.65	< 0.65
Mediano	≥ 0.90	≥ 0.75	≥ 0.60	< 0.60
Grande	≥ 0.85	≥ 0.70	≥ 0.55	< 0.55

CONCLUSIÓN

A partir de las métricas evaluadas podemos determinar el nivel de calidad de un producto y las áreas que requieren mejora para elevar la calidad total.