Ingeniería en computación
Universidad de La Serena
II Semestre, 2020
Versión número 1

Modelo de calidad

AIPA

Miércoles 21 de Octubre

Integrantes:
Yair Gallardo
Norton Irarrázabal
Sebastian Rojas

Docente
Guillermo Leyton.

Propósito

Este documento tiene como finalidad generar el modelo de calidad del software AIPA, teniendo como soporte teórico para su fabricación:

- Modelos de calidad para requisitos no funcionales como McCall, ISO 9126, ISO 25000.
- Métricas aportadas por Kesh, Mody, Piattini (tradicionales) y Mood, Chindamber & Kererer, Lorenz & kidd (orientados a objetos)

Requisitos funcionales

Los especificados en el ERS.

Requisitos no funcionales

Los factores de calidad determinados para AIPA son los siguientes:

Adecuación funcional: Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Completitud	Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario.	Imprescindible.	Las funciones del software deben cumplir con las tareas del usuario, ya que el no cumplimiento de sus funciones supone el fracaso del software.
Corrección	Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.	Parcial.	Si bien a nivel lógico debe funcionar correctamente las salidas no siempre serán correctas, ya que este es un software que aprende con el tiempo.
Pertinencia	Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos del usuario.	Imprescindible.	Las funciones del software deben poseer un único objetivo y ser coherentes con el fin de no distraer o quitar de contexto al usuario.

Eficiencia: Esta característica representa el desempeño relativo bajo ciertas características.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Comportamiento temporal	Los tiempos de respuesta y procesamiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas.	Imprescindible.	Las respuestas deben ser rápidas con el fin de no entorpecer la jugabilidad del usuario ni las dinámicas del juego.

Usabilidad: Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Inteligibilidad	Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.	Imprescindible.	El software debe ser enfocado a responder las necesidades que el usuario tenga, siempre y cuando pertenezcan al contexto en que se desenvuelve el software que en este caso es un juego de ajedrez capaz de brindar asistencia al usuario.
Aprendizaje	Capacidad del producto que permite al usuario aprender su uso.	Imprescindible.	Requiere ser intuitivo para el usuario, con la finalidad aprender rápidamente las dinámicas de su uso.
Operabilidad	Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.	Imprescindible.	Requiere ser intuitivo para el usuario con el fin de no complicar su experiencia.
Protección frente a errores	Capacidad del sistema para proteger a los usuarios de hacer errores.	Imprescindible.	El software debe ser capaz de evitar que el usuario tenga la posibilidad de generar errores durante su uso, con la finalidad de no entorpecer su uso ni experiencia de usuario.
Estética	Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.	Parcial.	Si bien la estética es importante, esta no debe ser totalmente pulcra, es decir no será un arreglo por consola, pero tampoco un juego con interfaz triple A/3D. El requisito es conseguir que no sea desagradable o que cause rechazo por parte del usuario.

Fiabilidad: Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Madurez	Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.	Imprescindible	Las funcionalidades del software deben responder de forma correcta, con la finalidad de no entorpecer la jugabilidad.
Disponibilidad	Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.	Imprescindible	El software debe encontrarse disponible para el usuario frente a toda condición normal, ya que este no debe requerir de elementos externos para su uso.
Tolerancia a fallos	Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.	Parcial.	El software debe ser capaz de funcionar correctamente posterior a un fallo evitando cierres inesperados.

Seguridad: Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Integridad	Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.	Imprescindible	De suma importancia ya que la modificación de los componentes del software por parte de terceros no autorizados generará que este deje de funcionar correctamente.

Mantenibilidad: Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Modularidad	Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.	Imprescindible	Importante para el software ya que debe tener bajo acoplamiento para evitar que la modificación de algunos de sus módulos afecte al resto.
Analizabilidad	Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.	Imprescindible	Importante para el software en caso de generar fallas debe ser informativo tanto para el equipo de desarrollo como para el usuario. Permitiendo trazabilidad.
Capacidad de ser probado	Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.	Imprescindible	Debe de ser posible aplicarle al software pruebas para poder determinar su correcto funcionamiento, contribuyendo a que los desarrolladores sean capaz de visualizar que el software cumple con sus funciones.

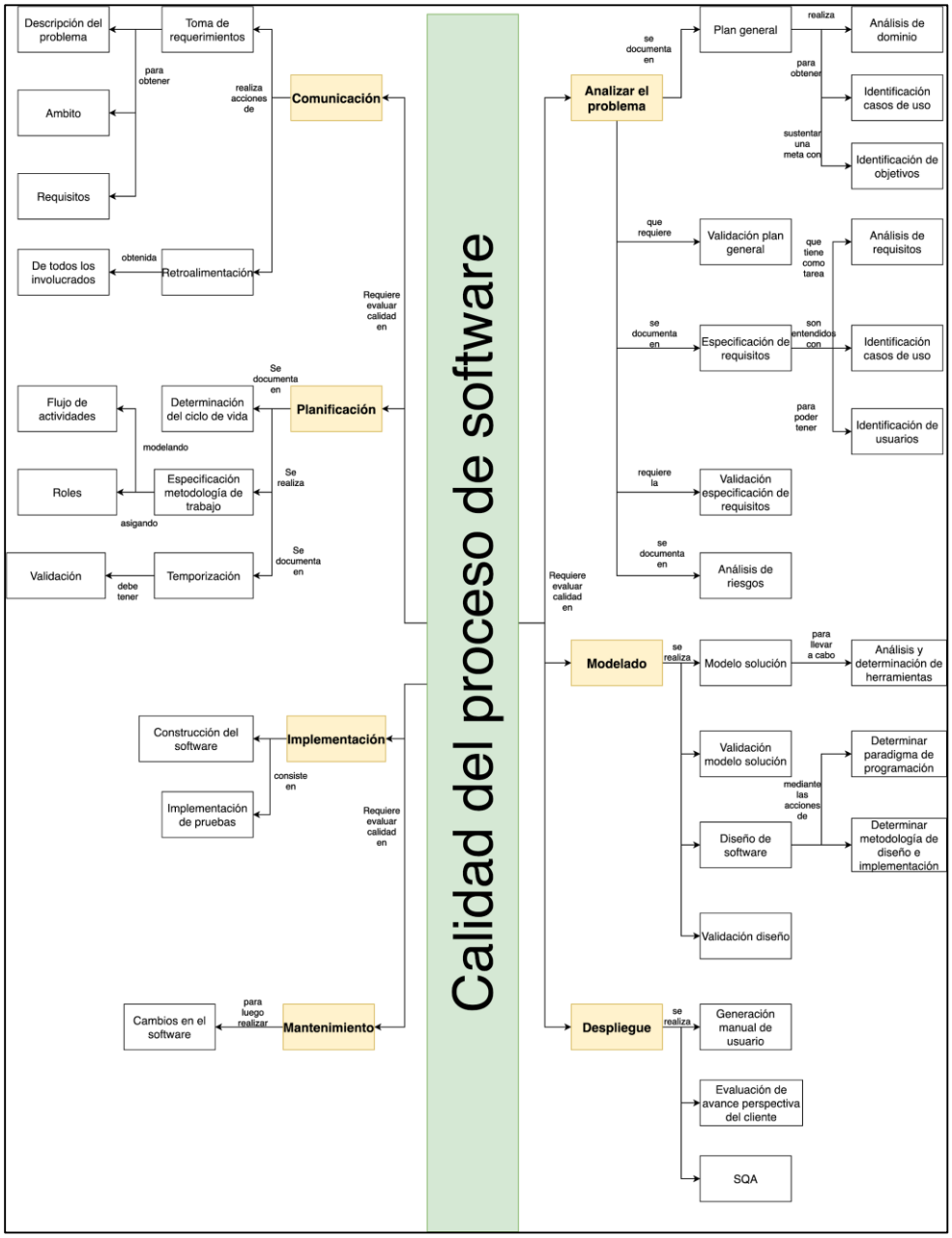
Portabilidad: Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otros.			
Subatributo	**Descripción**	**Importancia**	**Argumento**
Facilidad de instalación	Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.	Imprescindible	El software requiere ser de fácil instalación con la finalidad de facilitar su distribución en usuarios sin alto conocimiento en software y computadoras.

También debemos tomar en cuenta la perspectiva/Interés de los involucrados frente a cada uno de los atributos determinados (Internos/Externos).

Interesados	Atributos de calidad
Grupo de desarrollo	• Completitud. • Corrección. • Comportamiento temporal. • Protección frente a errores. • Integridad. • Modularidad. • Analizabilidad. • Capacidad de ser probado
Usuario	• Completitud. • Corrección. • Pertinencia. • Comportamiento temporal. • Inteligibilidad. • Aprendizaje. • Operabilidad. • Protección frente a errores. • Estética. • Madurez. • Disponibilidad. • Tolerancia a fallos. • Integridad. • Facilidad de instalación.
Cliente (Guillermo Leyton)	• Todos.

Comentado [1]: mejorar

Calidad del proceso



Métricas de calidad del proceso

Para medir la calidad del proceso se plantean una serie de preguntas que tienen como objeto de métrica la Tasa que se describe de la siguiente forma:

Tipo de métrica	Valor	Valor normalizado
Tasa	0 <= x <0.25	1
	0.25 <= x <0.5	2
	0.5 <= x <0.75	3
	0.75 <= x <1	4
	x = 1	5

Comunicación (CN)

1. ¿Se hizo toma de requerimientos?
    a. ¿Se describió el problema?
    b. ¿Se definió el ámbito?
    c. ¿Se definieron los requisitos?
2. ¿Se hizo retroalimentación?
    a. ¿Por todos los involucrados?

Analizar el problema (AP)

1. ¿Se realizó plan general?
    a. ¿Se realizó análisis del dominio?
    b. ¿Se identificaron los casos de uso?
    c. ¿Se identificaron los objetivos?
    d. ¿Se documento?
    e. ¿Se validó el plan general generado?
2. ¿Se realizó especificación de requisitos?
    a. ¿Se analizaron los requisitos?
    b. ¿Se analizaron los casos de uso?
    c. ¿Se Identificaron los usuarios?
    d. ¿Se documento?
    e. ¿Se hizo validación?
3. ¿Se analizaron los riesgos?

Planificación (PL)

1. ¿Se realizó ciclo de vida?
2. ¿Se definió la metodología de trabajo del grupo de desarrollo?
    a. ¿Se documentó el flujo de actividades del grupo de desarrollo?
    b. ¿Se designaron roles dentro del grupo de desarrolladores?
3. ¿Se realizó una temporización?
    a. ¿La temporización realizada fue validada?

Modelado (MOD)

1. ¿Se generó un modelo para la solución de la problemática?
    a. ¿Se realizó un análisis y determinación de herramientas?
    b. ¿Se validó el modelo generado?
2. ¿Se generó un diseño del software?
    a. ¿Se estableció un paradigma de programación adecuado?
    b. ¿Se estableció una metodología de diseño adecuada?
    c. ¿Se realizó una implementación adecuada?
    d. ¿El diseño fue validado?

Implementación (IMP)

1. Construcción de software
2. Implementación de pruebas

Despliegue (DES)

1. ¿Se generó un manual de usuario?
2. ¿Se realizaron evaluaciones de avance?
3. ¿Se realizaron revisiones SQA?

Mantenimiento (MAN)

1. ¿Se desarrolló el software respetando el diseño?
2. ¿Se desarrolló el software velando por la cohesión de cada uno de sus módulos?
3. ¿Se desarrolló el software velando por el bajo acoplamiento?

Enfoque ontológico

Es fundamental para hacer un buen modelo de calidad considerar el enfoque ontológico proporcionado por las métricas de Kesh[1].

Estructura: Son las relaciones que asocian las entidades del modelo.

- **CO1: Adecuación al problema**: Se refiere a que tanto el modelo refleja el problema que se quiere solucionar con la implementación de este.
- **CO2: Solidez**: Hace referencia a que tanto el modelo sigue con los principios técnicos del diseño (buenas prácticas, orientado al diseñador).
- **CO3: Consistencia**: Refleja el nivel de contradicción que tenga el modelo E/R, es decir, que tan estable y coherente es el producto de software.
- **CO4: Concisión**: Nivel de redundancia que tiene el modelo E/R. Capacidad de ser conciso en el diseño.

Contenido: Los atributos de las entidades del modelo.

- **CO5: Compleción**: Nivel de relevancia de los atributos que pertenecen al modelo E/R. El modelo contiene todas las sentencias acerca del dominio que son correctas y relevantes.
- **CO6: Cohesión**: Es la cercanía de los atributos, es decir, se busca que estos estén relacionados.
- **CO7: Validez**: Hace referencia a que los atributos sean válidos dentro del modelo E/R. Corresponde con las necesidades de negocio de clientes y usuarios.

Factor de calidad o de comportamiento: Adecuación funcional.

	Completitud	Corrección	Pertinencia
Estructura			
Adecuación al problema	X		X
Solidez		X	
Consistencia	X	X	X
Concisión			X
Contenido			
Compleción	X		X
Cohesión			X
Validez	X		X

[1] "Evaluating the quality of entity relationship models." https://sci-hub.do/10.1016/0950-5849(96)81745-9 . Fecha de acceso 21 oct.. 2020.

Factor de calidad o de comportamiento: Eficiencia.

	Comportamiento temporal
Estructura	
Adecuación al problema	
Solidez	
Consistencia	
Concisión	X
Contenido	
Compleción	
Cohesión	
Validez	

Factor de calidad o de comportamiento: Usabilidad.

	Inteligibilidad	Aprendizaje	Operabilidad	Protección frente a errores	Estética
Estructura					
Adecuación al problema					
Solidez				X	
Consistencia				X	
Concisión			X		
Contenido					
Compleción	X			X	
Cohesión	X	X	X		
Validez	X	X			X

Factor de calidad o de comportamiento: Fiabilidad.

	Madurez	Disponibilidad	Tolerancia a fallos
Estructura			
Adecuación al problema			
Solidez	X	X	X
Consistencia	X	X	X
Concisión			
Contenido			
Compleción	X		
Cohesión			
Validez	X		

Factor de calidad o de comportamiento: Seguridad.

	Integridad
Estructura	
Adecuación al problema	X
Solidez	X
Consistencia	
Concisión	
Contenido	
Compleción	X
Cohesión	
Validez	

Factor de calidad o de comportamiento: Mantenibilidad.

	Modularidad	Analizabilidad	Capacidad de ser probado
Estructura			
Adecuación al problema			
Solidez	X	X	X
Consistencia	X	X	X
Concisión	X	X	X
Contenido			
Compleción			
Cohesión	X	X	X
Validez			

Factor de calidad o de comportamiento: Portabilidad

	Facilidad de instalación
Estructura	
Adecuación al problema	X
Solidez	
Consistencia	
Concisión	
Contenido	
Compleción	
Cohesión	
Validez	

Fórmulas para el cálculo de la calidad del software

- Fórmula para la Calidad del software:

$$Q = \sum_{i=1}^{N} W_i * F_i$$

- Fórmula para factores de calidad (o de comportamiento):

$$F_i = \frac{\sum_{j=1}^{M_i} P_{ij} * S_{ij}}{M_i}$$

- Fórmula para los subfactores de calidad (o componentes de comportamiento):

$$S_{ij} = \frac{\sum_{k=1}^{L_{ij}} O_k}{L_{ij}}$$

- Fórmula para el cálculo de calidad del software ampliada (reemplazando variables compuestas)

$$Q = \{ \sum_{i=1}^{N} W_i * [\frac{\sum_{j=1}^{M_i} P_{ij} * (\frac{\sum_{k=1}^{L_{ij}} O_k}{L_{ij}})}{M_i}] \}$$

donde:

Nivel 3.

- Q : $Cruce\ del\ modelo\ y\ factores\ de\ calidad$
- N : $Número\ total\ de\ factores\ de\ calidad\ = 7$
- W_i : $Peso\ del\ factor\ de\ calidad\ i$
- F_i : $Valor\ del\ factor\ de\ calidad\ i$

Nivel 2.

- M_i : $Número\ total\ de\ subfactores\ del\ factor\ i$
- P_{ij} : $Valor\ independiente\ del\ sub\ factor\ de\ calidad\ j\ del\ factor\ de\ calidad\ i$
- S_{ij} : $Sumatoria\ de\ los\ valores\ ontológicos\ implicados\ en\ el\ subfactor\ de\ calidad\ j\ del\ factor\ de\ calidad\ i$

Nivel 1.

- L_{ij} : $Número\ total\ de\ Componentes\ ontológicos\ implicados\ en\ el\ subfactor\ j\ del\ factor\ i$

- O_k : $Valor\ del\ componente\ ontológico\ k$

Factores de calidad	Peso asociado
Adecuación funcional	0.184
Eficiencia	0.148
Usabilidad	0.184
Fiabilidad	0.142
Seguridad	0.1
Mantenibilidad	0.142
Portabilidad	0.1

Evaluación de los componentes ontológicos

Para medir la calidad de los factores de calidad se plantean una serie de preguntas que tienen como objeto de métrica la Tasa que se describe de la siguiente forma:

Tipo de métrica	Valor	Valor normalizado
Tasa	0 <= x <0.25	1
	0.25 <= x <0.5	2
	0.5 <= x <0.75	3
	0.75 <= x <1	4
	x = 1	5

Adecuación al problema (O1)

- ❑ Entrevista con usuario (valoración 1 a 5)
  - ❑ ¿Considera que el modelo que estamos proponiendo se adecua a su problema?
  - ❑ ¿El modelo considera los aspectos relevantes de su problema?
  - ❑ ¿El modelo toma todo el dominio del problema?

Solidez (O2)

- ❑ Entrevista con el equipo técnico (valoración 1 a 5)
  - ❑ ¿Se aplicó diseño por contrato?
  - ❑ ¿Se utilizaron elementos de los principios SOLID y/o GRASP?
  - ❑ ¿Se identificaron malas prácticas en el modelo (DRY, elementos mejorables)?

Consistencia (O3)

- ❑ También se evalúa con valor 1 a 4, y utiliza las siguientes fórmulas:
  - ❑ ¿El modelo presenta contradicciones?.
  - ❑ Si presenta inconsistencias, ¿cuantas presenta y a que relaciones afecta?.
  - ❑ ¿El modelo es coherente con el producto software?
  - $O_3 = 5 - D$
  - $D = \frac{R}{4n}$

  Donde:
  - $R = número\ de\ inconsistencias$
  - $n = número\ de\ relaciones\ en\ el\ modelo$

Consición (O4)

- ❑ Valoración 0 a 5. Donde:
  - ❑ ¿El modelo es conciso con el diseño?
  - ❑ ¿El modelo cumple con la meta propuesta?
  - ❑ ¿El modelo presenta entidades y relaciones repetitivas?
  - ❑ Fórmula: $M\{ (\ \ {}^{n}c_2 - n)/ [\ \ {}^{n}c_2 - (n-1)] \}$

  Donde:
  - $n = número\ de\ entidades\ en\ el\ modelo$
  - ${}^{n}c_2 = número\ de\ relaciones\ en\ el\ modelo$
  - $M = número\ máximo\ de\ puntaje = 5$

Compleción (O5)

- ❑ Se definen un conjunto de consultas a realizarse en el modelo y se les asigna un valor de importancia a cada una. Se realizan las consultas y por cada falla se le resta a 5 el valor de importancia de aquella pregunta.
  - ❑ ¿Las entidades del modelo consideran todos los atributos necesarios para su representación?
  - ❑ ¿El modelo contiene todas las sentencias lógicas del dominio del problema?
  - ❑ ¿Estas sentencias son correctas?

Cohesión (O6)

- ❑ Considera:
  - ❑ ¿Los atributos de las entidades de su modelo están relacionados?

- ❑ ¿Presenta entidades con uno o más atributos no relacionados?
  - ❑ ¿Cuantas entidades presentan este problema?
- ❑ ¿Su modelo presenta entidades "Dios" (entidades que contienen una gran porción de todos los atributos involucrados en el modelo)?
- ❑ ¿Presenta entidades con múltiples comportamientos, encargados de tareas no relacionadas?
  - Si el contenido está formado por un solo atributo = 5
  - Si el contenido está formado por todos los atributos = 0
  - Fórmula: $\sum_{i=1}^{n} \frac{o_{6i}}{n}$, donde: $o_{6i} = M\{[n_e - n_p/(n_e - 1)]$

Donde:

$$o_{6i} = Entidad\ i$$
$$n_e = cantidad\ de\ atributos\ en\ la\ entidad\ o_{6i}$$
$$n_p = cantidad\ de\ atributos\ que\ componen\ al\ identificador\ de\ la\ entidad\ o_{6i}$$

Validez (O7)

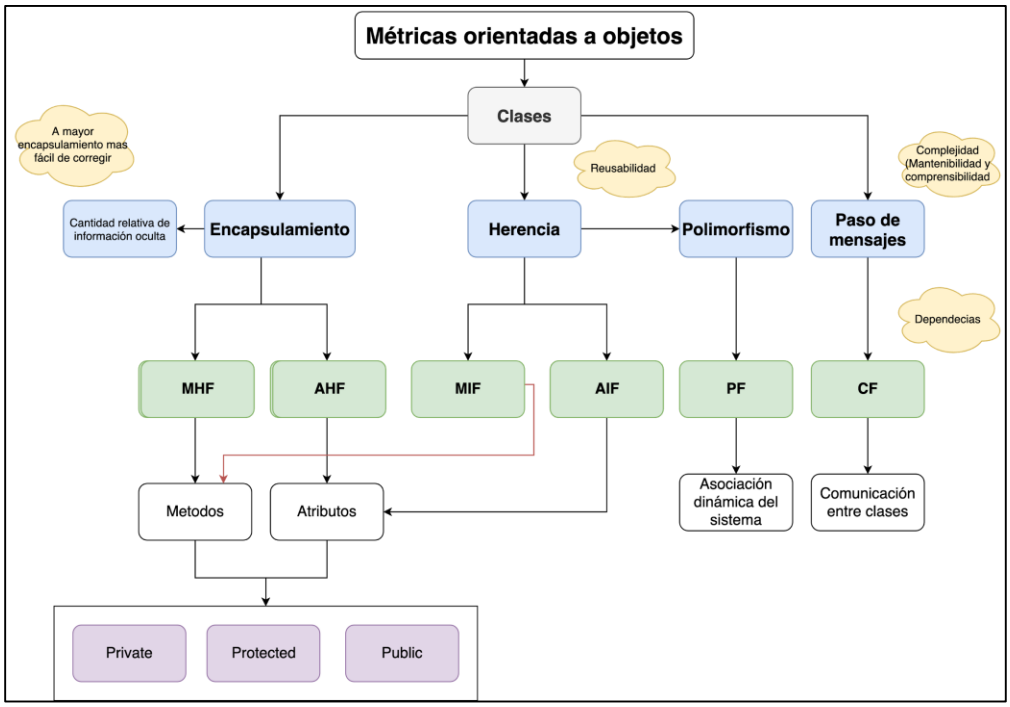
- ❑ ¿El modelo cumple con las reglas del negocio?
- ❑ ¿El modelo cumple con lo requerido por el cliente?
- ❑ ¿El modelo cumple con los requisitos de los usuarios?
- ❑ ¿Los atributos considerados son válidos? es decir, corresponden al modelo del mundo (entorno en el que se desenvuelve el problema con una determinada lógica) que se está tratando de resolver?
  - Todos los atributos son válidos para todas las entidades = 5
  - Todos los atributos no son válidos para todas las entidades = 0
  - Fórmula: $M\left(\frac{1-n_i}{\sum n_e}\right)$

Donde:

- $n_i = número\ total\ de\ entidades\ no\ válidas.$

Orientación a objeto

Para evaluar la orientación a objeto se utilizó el siguiente esquema, proporcionado por el modelo Mood[23] :



El cual consta de 6 factores definidos a continuación.

[2] (n.d.). The Quest for Software Components Quality - CTP. Recuperado el octubre 19, 2020, de http://ctp.di.fct.unl.pt/QUASAR/Resources/Papers/2002/2002_COMPSAC.pdf

[3] (n.d.). MEDICIÓN EN LA ORIENTACIÓN A OBJETOS. Recuperado el octubre 19, 2020, de http://www.cc.uah.es/drg/b/RodHarRama00.pdf

Method Hiding Factor (MHF) - *Factor de encapsulamiento de los métodos*

Relación entre los métodos definidos como protegido-privados, y el número total de métodos del software.

Fórmulas:

$$MHF = \frac{\sum_{i=1}^{TC}\sum_{m=1}^{Md(C_i)}\left(1 - V(M_{mi})\right)}{\sum_{i=1}^{TC} Md(C_i)}$$

$$V(M_{mi}) = \frac{\sum_{j=1}^{TC} es\_visible(M_{mi}, C_j)}{TC - 1}$$

$$es\_visible(M_{mi}, C_j) = \begin{cases} 1 \Leftrightarrow j \neq i \wedge C_j \text{ puede llamar a } M_{mi} \\ 0 \qquad \text{caso contrario} \end{cases}$$

$Es\ decir: \forall C_i\ , (M_{mi}\ =\ 0\ si\ puede\ usarse\ desde\ otra\ clase) \vee (M_{mi} = 1)$[4]

Donde:

- $MHF: Suma\ de\ las\ invisibilidades\ de\ todos\ los\ métodos\ definidos\ en\ todos\ las\ clases\ del\ software.$
- $TC\ :\ Es\ el\ número\ total\ de\ clases\ en\ el\ software$
- $M_d(C_i): Cantidad\ de\ métodos\ declarados\ (no\ heredados)\ dentro\ de\ una\ clase\ C_i$
- $V(M_{mi})\ :\ Visibilidad\ \%\ del\ total\ de\ clases\ desde\ que\ el\ método\ M_{mi}\ de\ la\ clase\ C_i\ es\ visible$

Attribute Hiding Factor (AHF) - *Factor de encapsulamiento de los atributos*

Relación entre los atributos definidos como protegidos o privados y el número total de atributos. Se propone como una métrica para la encapsulación.

Fórmulas:

$$AHF = \frac{\sum_{i=1}^{TC}\sum_{m=1}^{Ad(C_i)}\left(1 - V(A_{mi})\right)}{\sum_{i=1}^{TC} Ad(C_i)}$$

$$V(A_{mi}) = \frac{\sum_{j=1}^{TC} es\_visible(A_{mi}, C_j)}{TC - 1}$$

[4] Vmi toma un valor distinto cuando el método es protegido. Revisar pie de página anterior.

$$es\_visible(A_{mi}, C_j) = \begin{cases} 1 \Leftrightarrow j \neq i \wedge C_j \text{ puede llamar a } A_{mi} \\ 0 \quad \text{caso contrario} \end{cases}$$

$$Es\ decir: \forall C_i\ , (A_{mi}\ =\ 0\ si\ puede\ usarse\ desde\ otra\ clase) \vee (A_{mi} = 1)$$

Dónde:

- $MHF: Suma\ de\ las\ invisibilidades\ de\ todos\ los\ atributos\ definidos\ en\ todas\ las\ clases\ del\ software.$
- $TC: Es\ el\ número\ total\ de\ clases\ en\ el\ software$
- $A(C_i): Cantidad\ de\ atributos\ declarados\ (no\ heredados)\ dentro\ de\ una\ clase\ C_i$
- $V(A_{mi}): Visibilidad\ \%\ del\ total\ de\ clases\ desde\ que\ el\ atributo\ M_{mi}\ de\ la\ clase\ C_i\ es\ visible$

Method Inheritance Factor (MIF) - *Factor de herencia de los métodos*

Proporción de la suma de todos los métodos heredados en todas las clases entre el número total de métodos (localmente definidos más los heredados) en todas las clases. Busca expresar el nivel de reusabilidad junto a delimitar el número de elementos a testear.

Fórmulas:

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

donde:

$$M_a(C) = M_d(C) + M_i(C)$$

y:

$M_d(C_i)$ es el número de métodos declarados en una clase

$M_a(C_i)$ es el número de métodos que pueden ser invocados en relación a C_i.

$M_i(C_i)$ es el número de métodos heredados (y no redefinidos C_i).

TC es el número total de clases en el sistema.

Attribute Inheritance Factor (AIF) - Factor de herencia de los atributos

Proporción del número de atributos heredados entre el número total de atributos. Busca expresar el nivel de reusabilidad.

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

donde:

$$A_a(C) = A_d(C) + A_i(C)$$

y:

$A_d(C_i)$ es el número de atributos declarados en una clase.
$A_a(C_i)$ es el número de atributos que pueden ser invocados asociados a C_i.
$A_i(C_i)$ es el número total de atributos heredados (y no redefinidos) en C_i.
TC es el número total de clases en el sistema.

Polymorphism Factor (POF) - *Factores de polimorfismo*

Número de métodos heredados redefinidos dividido por el máximo número de situaciones polimórficas distintas. Es una medida de polimorfismo y una medida indirecta de la asociación dinámica en el software.

Fórmulas:

$$PF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

Donde:

$$M_d(C_i) = M_n(C_i) + M_o(C_i)$$

y,

$M_n(C_i)$ es el número de métodos nuevos.
$M_o(C_i)$ es el número de métodos redefinidos.
$DC(C_i)$ es el número de descendientes de C_i.
TC es el número total de clases en el sistema.

Coupling Factor (COF) - *Factores de acoplamiento*

Indica la comunicación entre las clases. Busca servir como una medida del incremento de la complejidad, la cual reduce la encapsulación y reuso, y por tanto limitando la comprensibilidad y mantenibilidad de software

Fórmula:

$$CF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} es\_cliente(C_i, C_j)\right]}{TC^2 - TC}$$

Donde:

$$es\_cliente = \begin{cases} 1 \Leftrightarrow C_c \Rightarrow C_s \wedge C_c \neq C_s \\ 0 \quad caso\ contrario \end{cases}$$

La relación cliente-proveedor ($C_c \Rightarrow C_s$) representa que la clase cliente (C_c) contiene al menos una referencia no heredada de la clase proveedor (C_s).