UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSTGRADO

PROGRAMA DE MAESTRIA

MAESTRIA EN INFORMÁTICA

TESIS

SOFTWARE PARA LA ESTIMACION AUTOMÁTICA DE LA CABEZA HUMANA EN 3D MEDIANTE AAM Y POSIT

PRESENTADA POR:

RENZO APAZA CUTIPA

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA
MENCION EN INGENIERÍA DE SOFTWARE

PUNO, PERÚ

2015

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSTGRADO

PROGRAMA DE MAESTRIA

MAESTRIA EN INFORMÀTICA

TESIS

SOFTWARE PARA LA ESTIMACION AUTOMÁTICA DE LA CABEZA HUMANA EN 3D MEDIANTE AAM Y POSIT

PRESENTADA POR:

RENZO APAZA CUTIPA

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA
MENCION EN INGENIERÍA DE SOFTWARE

PUNO, PERÚ

2015

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSTGRADO

PROGRAMA DE MAESTRÍA

MAESTRÍA EN INFORMÁTICA

TESIS

SOFTWARE PARA LA ESTIMACION AUTOMATICA DE LA CABEZA HUMANA EN 3D MEDIANTE AAM Y POSIT

PRESENTADA POR:

RENZO APAZA CUTIPA

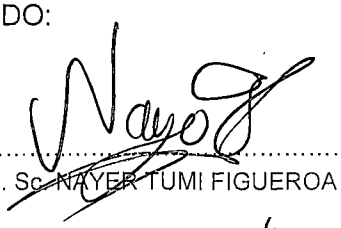
PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN

INFORMÁTICA MENCION EN INGENIERÍA DE SOFTWARE

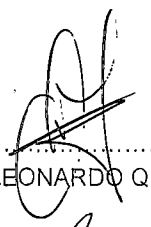
APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE



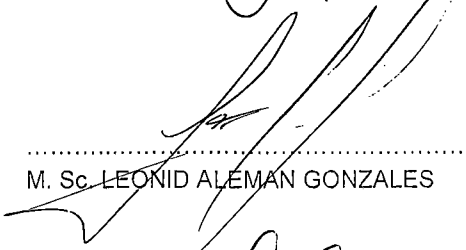
M. Sc. NAYER TUMI FIGUEROA

PRIMER MIEMBRO



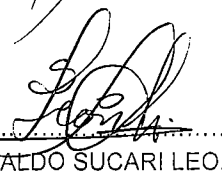
M. Sc. PEDRO LEONARDO QUISPE TICONA

SEGUNDO MIEMBRO



M. Sc. LEONID ALEMAN GONZALES

ASESOR DE TESIS



M. Sc. REYNALDO SUCARI LEON

Puno, 29 de Enero de 2015

DEDICATORIA

Dedico este trabajo a Dios nuestro creador por los dones y bendiciones que vierte sobre nosotros tus hijos, por permitir compartir mi vida con personas maravillosas y por guiar e iluminar mí camino.

A la persona que alegro mi ser y compartio mis sueños, Fiorella eres tú la compañera ideal, las más infinitas expresiones de gratitud por tu apoyo e incondicional confianza.

Para ti Mateo Jacobo la ilusión de mi vida, tu sonrisa es luz que ilumina y alegra mi corazón, la esperanza de poder ver y estar a tu lado aviva mis ideales, principios y los sentimientos más nobles que alguien pueda tener.

AGRADECIMIENTOS

A las autoridades de la Maestría en Informática de la Escuela de Post grado de la Universidad Nacional del Altiplano.

INDICE GENERAL

MARCO TEÓRICO

INDICE DE CUADROS

INDICE DE FIGURAS

INDICE DE ANEXOS

RESUMEN

La estimación de la posición de la cabeza es un paso clave para numerosas aplicaciones humano computador que utilizan el rostro como el seguimiento de ojos, reconocimiento de rostros y ecografías 4D para prenatales. Por ello en el presente trabajo se desarrolló un software para estimar en tiempo real, la posición de una cabeza humana en 3D. Para ello se realizó la identificación de los requerimientos de usuario, el análisis, diseño e implementación de los módulos. Se implementaron los algoritmos Active Apareance Model (AAM), POSIT para acoplar las mediciones realizadas en 2D a 3D y subdivisiones mediante la triangulación de Delaunay, que se usan como una estructura de malla. La implementación dio como resultado el software que encaja el modelo en 3D usando emparejamiento entre puntos del modelo, un seguidor de rostros para detección que produce 6 grados de libertad para estimación de posición, 3 grados para rotación y 3 grados para traslación. Finalmente se evaluó la calidad del software para la estimación de la cabeza humana basados en el ISO/IEC 9126, esta permitió determinar un conjunto de características que cumplen con los requisitos especificados por el usuario logrando una calidad interna del 81 % y una calidad externa del 85%. Además, la evaluación de la hipótesis General dio como resultado que el software tiene una eficiencia aceptable en la estimación de la posición de la cabeza humana mediante el uso de AAM y POSIT.

PALABRAS CLAVE: Modelos activos de apariencia, Modelos activos de forma, Transformada Afine, Triangulación de Delaunay, POSIT, Análisis de Componentes Principales,

ABSTRACT

The estimate of the position of the head is a key step for numerous human computer applications that using the face as eye tracking, face recognition and ultrasounds 4D for prenatals. Therefore in this paper was developed software to estimate in real time, the position of a human head in 3D. For this purpose the identification of user requirements was conducted, analysis, design and implementation of the modules. Were implemented the algorithms Active Appearance Model (AAM), POSIT for coupling measurements in 2D to 3D and subdivisions by Delaunay triangulation which are used as a mesh structure. Resulted implementing software, which fits the 3D model using the model match between points, a follower face detection that produces 6 degrees of freedom for position estimation, 3 degrees for rotation and 3 degrees for translation. Finally the quality software for estimating human head was performed based on the ISO / IEC 9126, this allowed to determine a set of features that meet the requirements specified by the user achieving a 81% internal quality and external quality was assessed 85%. Furthermore, evaluation of the hypothesis General gave resulted that software has an acceptable efficiency in estimating the position of the human head using AAM and POSIT.

KEYWORDS: Active Appearance Model, Active Shape Model, Affine Transformation, Delaunay Triangulation, POSIT, Principal Component Analysis,

INTRODUCCIÓN

El poder interpretar y caracterizar un rostro es una funcionalidad importante para la Interacción Hombre Maquina (Human Computer Interaction). El hecho de conocer la postura, orientación y posición de la cara, como la habilidad de reconocer identidades o expresiones en estas, son características que permiten la construcción de sistemas interactivos, como: Sistemas de reconocimiento de expresiones, estimación de estados de ánimo, aplicaciones de videoconferencia, identificación automática de identidad, simulación en ambientes virtuales, análisis de perfiles de personalidad, etc.

Debido al potencial de sus aplicaciones en diversos ámbitos el reconocimiento de expresiones es un ejemplo importante de técnicas de reconocimiento de rostros usadas en ambientes inteligentes.

En el presente trabajo el primer capítulo expone la problemática de la investigación objeto de estudio del presente, a continuación se describen y repasan los estudios previos realizados con relación a los temas del presente trabajo, luego se detallan los objetivos de la investigación necesarios para la implemetación del software de estimación de la cabeza basados en AAM y el algoritmo POSIT.

A continuación se prosigue con la exposición del segundo capítulo donde se trata el marco teórico de los temas más resaltantes entre estos los Modelos de forma activa, Modelos de apariencia activa, la técnica de Análisis de Componentes Principales, la triangulación de Delanuay, Warp transformation y POSIT – Pose from orthography and scaling with Iterations. Sigue luego con el

marco conceptual de los términos utilizados en la investigación. Y prosiguiendo este capítulo se formula la hipótesis de la investigación

El tercer capítulo prosigue con el detalle de la metodología que se utilizó en la investigación, también se determina el tipo y diseño de investigación, la metodología utilizada para evaluar la calidad del software desarrollado y por último se describe la población y la muestra de la investigación.

En el cuarto capítulo se exponen los resultados del desarrollo del software, se hace una exposición del análisis de requisitos, el análisis y diseño del software, la implementación y la evaluación de la calidad del software donde se usó como referencia la norma ISO/IEC 9126.

Ya al final del presente documento se detallan las conclusiones a las que se arribó como consecuencia del desarrollo del software, las recomendaciones correspondientes y los anexos.

CAPÍTULO I

PROBLEMÁTICA DE INVESTIGACIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

La capacidad de estimar la posición de la cabeza de una persona es una habilidad común de los seres humanos, pero presenta un desafío para las aplicaciones de visión por computador. Las aplicaciones que utilizan el rostro como pieza clave (tales como la identificación biométrica de personas mediante rostros, la interacción humano computador, videojuegos y otros) se ven afectadas por la posición de la cabeza dentro de las imágenes debido a que, al no estar posicionadas de la forma esperada genera un tipo de ruido por oclusiones no previstas en la entrada a estos sistemas; estas aplicaciones se ven afectadas por la variación de la posición del rostro aumentado de esta manera la tasa de falso positivo y de falsos rechazos en distintas tareas ante la presencia de un rostro en una determinada imagen.

Las dificultades a superar son muchas y variadas. Las imágenes de los rostros humanos por lo regular muestran un alto grado de variabilidad en la

forma y la textura. Estas variaciones fijadas en la apariencia se deben a las diferencias entre las imágenes de los individuos, las deformaciones en la expresión facial, la postura y los cambios de iluminación. Así la información extraída debe tener una dimensión acotada, ya que para procesar una imagen completa se requiere un enorme esfuerzo de cálculo.

Debido al hecho que la estimación de la posición de la cabeza de un sujeto juega un rol primordial en diversas aplicaciones la presente investigación busca resolver la siguiente interrogante:

¿Cuál es la eficiencia del software en la estimación automática de la posición de la cabeza mediante el uso de Active Apareance Model y POSIT?

1.2. OBJETIVO

1.2.1. OBJETIVO GENERAL

Determinar la eficiencia del software en la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT.

1.2.2. OBJETIVO ESPECIFICO

- Identificar los requerimientos para el desarrollo del software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT.
- Analizar y diseñar el software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT.

- Implementar el software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT.
- Evaluar la efectividad del software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT basado en las métricas ISO/IEC 9126.

1.3. HIPOTESIS

1.3.1. HIPOTESIS GENERAL

El software tiene una eficiencia aceptable en la estimación automática de la posición de la cabeza mediante el uso de Active Apareance Model y POSIT.

1.3.2. HIPOTESIS ESPECÍFICA

- Los requerimientos identificados permiten el desarrollo del software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT.
- El análisis y diseño facilita el desarrollo del software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT.
- La evaluación dela eficiencia del del software para la estimación automática de la posición de la cabeza mediante el uso de AAM y POSIT basada en ISO/IEC 9126 permite determinar un conjunto de características que cumplen con los requisitos de usuario.

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

- T. F. Cootes, C.J. Taylor, D.H. Cooper, & J.Graham, *Active Shape Models – Their Training and Application.*

  La investigación describe un método para la construcción de un modelo aprendiendo los patrones de variabilidad del conjunto de entrenamiento de las imágenes anotadas. Estos modelos pueden ser usados para buscar imágenes en un algoritmo iterativo análogo al empleado por Active Contours Models con la diferencia que el modelo de forma activa solo puede modificar el modelo para ajustar a la data de forma cosistente basado en el conjunto de entrenamiento (Cootes, Taylor, Cooper, & Graham, 1995).

- T.F. Cootes, G.J. Edwards & C.J. Taylor, *Active Aparece Models.*

  Presentan el metodo Active Apareance Model – Modelo de Apariencia Activa. El método contiene un modelo estadístico de la forma y niveles de

gris de la apariencia de algún objeto de interés. Para encontrar una imagen miden los errores residuales y usan el modelo para predecir cambios a los parámetros, se obtienen buenos resultados en pocas iteraciones, (Cootes, Edwards, & Taylor, Active Appearance Models, 1998).

- Pedro Alexandre Dias Martins, *Active Appearance Models for Facial Expression Recognition and Monocular Head Pose Estimation.*

  Describe el trabajo realizado con modelos de rostro con respeto al análisis de la expresión facial y estimación de la postura. El trabajo está compuesto por tres módulos principales, Active Appearance Model (AAM), el análisis de expresión facial y reconocimiento (FEAR) y el modulo Monocular Head Pose Estimation. (Dias Martins, 2008).

- Daniel F. DeMenthon y Larry S. Davis, *Model-Based Object Pose in 25 Lines of Code*

  Describe un método para encontrar la postura. de un objeto desde una simple imagen, asume que se puede detectar y coincidir en la imagen cuatro o más puntos característicos no coplanares del objeto y se conoce la geometría relativa del objeto. El método combina dos algoritmos; el primero POS (Pose from Orthography and Scaling) aproxima la proyección perspectiva con una proyección ortográfica escalada y encuentra una matriz de rotación y el vector de traslación del objeto resolviendo un sistema lineal; el segundo algoritmo POSIT (POS con iteración), utiliza en su bucle de iteración la aproximación de la postura hallada con POS con el fin de calcular una mejor proyección ortográfica

escalada de los puntos característicos, para a continuación aplicar POS a estas proyecciones en lugar de las proyecciones de la imagen original.

- Ali Md. Haider y Toyohisha Kaneko, *Realistic 3D Head Modeling from Video Captured Images and CT Data*

  Usando una técnica de esculpir basado en imágenes de la cabeza rotados proponen un método de reconstrucción de la forma del cabello así como las secciones cóncavas y ocluidas de la intersección del cabello y la piel. Para obtener un registro automático completo es necesaria que el cabello no cubra las orejas debido a que este es un punto de referencia (Haider & Kaneko, 2000).

- Vel Murugan M & Sam Mathews M, *2D and 3D Active Shape Model with SURF Algorithm for Object Retrieval.*

  Presenta un modelo que usa simultáneamente el Modelo de Forma Activa en ambientes en 3D y el método con el algoritmo SURF (Speeded Up Robust Features) para mejorar el sistema de recuperación de imagen. Agregan el algoritmo SURF al modelo de forma activa en 3D para detectar y describir características locales en imágenes. El método da buenos resultados de recuperación y es robusto comparado a otros métodos (Murugan M. & Mathews M., 2013).

- Jacob Whitehill & Javier R. Movellan, *A Discriminative Approach to Frame-by-Frame Head Pose Tracking.*

  Realizan un acercamiento cuadro a cuadro del seguimiento de la postura de la cabeza que es robusto a un amplio rango de iluminaciones y apariencia facial. Los resultados sobre una diversa base de datos de imágenes indican que un array de discriminativos clasificadores de rango

y postura, integrados usando regresión lineal, pueden producir niveles de precisión cercanos a los humanos usando software de etiquetado grafico en 3D (Whitehill & Movellan, 2008).

- Andreas Launila & Josephine Sullivan, *Contextual Features for Head Pose Estimation in Football Games.*

  Los autores exploran los beneficios de usar características contextuales para la estimación de la posición de la cabeza en juegos de football. Las características contextuales son derivadas del conocimiento de la posición de todos los jugadores y combinadas con características basadas en las imágenes derivadas de mediciones de baja resolución. Usando selección de características y combinación de técnicas, muestran que las características contextuales pueden ayudar a la estimación de la postura de la cabeza en juegos de football y potencialmente ser un importante complemento a las características basadas en imágenes tradicionalmente utilizadas.

- Roberto Valenti, Nicu Seve & Theo Gevers, *Combining Head Pose and Eye Location Information for Gaze Estimation.*

  Proponen un esquema para combinar la posición de la cabeza y la información de la posición del ojo para obtener una mejor estimación de la mirada. Para este fin, la matriz de transformación obtenida para la posición de la cabeza es utilizada para normalizar las regiones de los ojos, sucesivamente, la matriz de transformación generada por la ubicación del ojo hallado es utilizado para corregir el procedimiento de la estimación de la postura. El esquema está diseñada para mejorar la precisión de la estimación de la ubicación del ojo. Particularmente en videos de baja

resolución, para extender el rango operativo dela ubicación del ojo, y para mejorar la precisión de l rastreador de la postura de la cabeza. Estas mejoras en las estimaciones son entonces combinadas para obtener un nuevo sistema para la estimación de la mirada, el cual usa ambos ubicación del ojo y la información de la cabeza para mejorar la estimación de la mirada. De los resultados experimentales se deriva que el esquema unificado propuesto mejora la precisión de la estimación del ojo de un 16% a 23%. Aún más extiende considerablemente su rango operativo por mas del 15°. La precisión del rastreador de la cabeza es mejorada de un 12% a un 24% (Valenti, Sebe, & Gevers, 2012).

- Hiromasa Yoshimoto & Yuichi Nakamura, *Free-Angle Head Pose Tracking Base don Online Shape Acquisition*.

  Proponen un modelo para la adquisición de la forma y postura de la cabeza. Este modelo incrementalmente adquiere la forma de la superficie 3D de una cabeza como un objeto rígido simple. El modelo incluye no solo la región de la superficie frontal del rostro sino también las regiones laterales del rostro, orejas, cabellos y algunos objetos extraños como lentes si presentan. La forma 3D de la cabeza es una colección de una multitud de apariencias de la cabeza tomadas de varios puntos de vista. Sus modelos no sufren limitaciones con respecto al ángulo de orientación y posición de la cabeza. Formalizan el acercamiento como una combinación de 2 procesos en tiempo real: la adquicion de la forma de la superficie de la cabeza de forma incremental en 3D y seguimiento de su postura utilizando seis grados de libertad para la posición y la orientación.

2.2. BASE TEÓRICA

2.2.1. MODELO ACTIVO DE FORMA (ACTIVE SHAPE MODEL - ASM

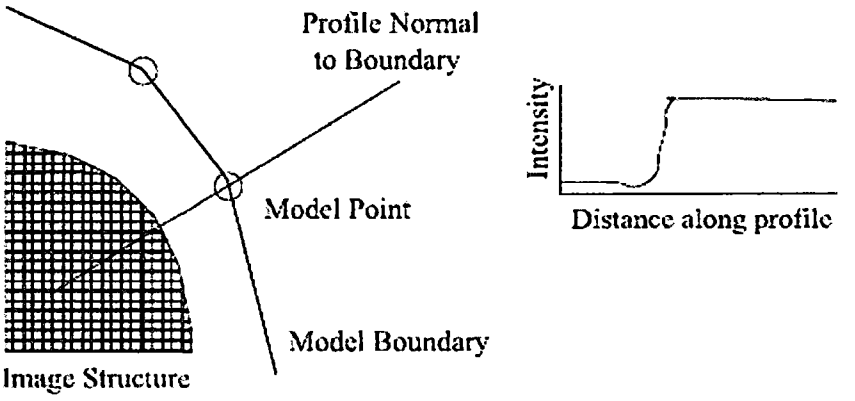
El Modelo Activo de Forma puede ser visto como un ejemplo especifico de un modelo local con restricciones (CLM) una clase de algoritmo que realiza una búsqueda local para cada característica (basado en un conjunto independiente de modelos de textura aprendidos) entonces ajustar un modelo de forma aprendido al conjunto de coincidencias locales. Localizar la susceptibilidad a mínimos locales, sin embargo, ha sido dirigido para forzar varias modificaciones a las medidas de coincidencia y algoritmos de búsqueda.

El ASM coincide un modelo a una imagen no vista anteriormente alternando búsquedas locales de características para maximizar la bondad de ajuste y regularizando las formas situados filtrando las falsas coincidencias causadas por el ruido de los datos.

Dado un conjunto de valores de los parámetros de forma, **b**, y parámetros de posición, **t**, se puede definir la forma del objeto dentro del marco de la imagen. Si se define la medida de que tan bien los parámetros dados explican la data de la imagen observada, se pueden encontrar mejores valores para los parámetros buscando en regiones locales alrededor de cada punto característico para encontrar ubicaciones de características alternativas que coincidan con el modelo lo más próximo posible. En general, se puede modelar la apariencia con un parche 2D entrado en la ubicación de la característica y buscar una region

2D de interés alrededor del estimado actual para una mejor coincidencia.

FIGURA 1
EN CADA PUNTO MODELO, MUESTREO A LO LARGO DE PERFIL NORMAL A LA FRONTERA.



En el caso específico de ASM se reduce la demando computacional consultando a lo largo de los perfiles lineales 1D que pasan atreves de cada punto modelo y es normal a las fronteras del modelo (FIGURA 1). Si se asume que las fronteras del modelo corresponden a un borde, el borde más fuerte a lo largo del perfil sugiere una nueva ubicación para el punto modelo. Los puntos modelo, sin embargo, no siempre son hallados sobre el borde más fuerte en la localización, en cambio pueden estar asociados con un borde secundario débil, o alguna otra estructura en la imagen, asi en lugar de aprender del conjunto de entrenamiento observado para la imagen objetivo.

Un método popular es construir un modelo estadístico de la estructura de los niveles de gris del perfil, normal a la frontera en el conjunto de entrenamiento. Supóngase para un punto dado se muestree a lo largo de un perfil *k* pixels a cada lado del modelo el

punto en el iesima imagen de entrenamiento. Entonces se tiene $2k + 1$ muestras que pueden ser colocadas en un vector gi. Para evitar los efectos de una constante compensación en las intensidades se muestrea a lo largo del perfil derivable más que los valores absolutos de los niveles de gris. Similarmente compensa por cambios en contraste dividiendo a través de la suma de valores de los elementos absolutos como:

$$\mathbf{g}_i \rightarrow \frac{1}{\sum_j |g_{ij}|}\mathbf{g}_i.$$

Esto se repite para toda imagen de entrenamiento hasta obtener un conjunto de muestras normalizadas, {gi}, cuyas distribuciones se pueden modelar. Si se asume que estas muestras de perfiles tienen una distribución Gaussiana multivariable, por ejemplo, se puede construir un modelo estadístico de los niveles de gris del perfil calculando su media, g, y covarianza, Sg. La calidad de ajuste de una nueva muestra, gs, al modelo es entonces dado por la distancia de Mahalanobis de la muestra para la media del modelo:

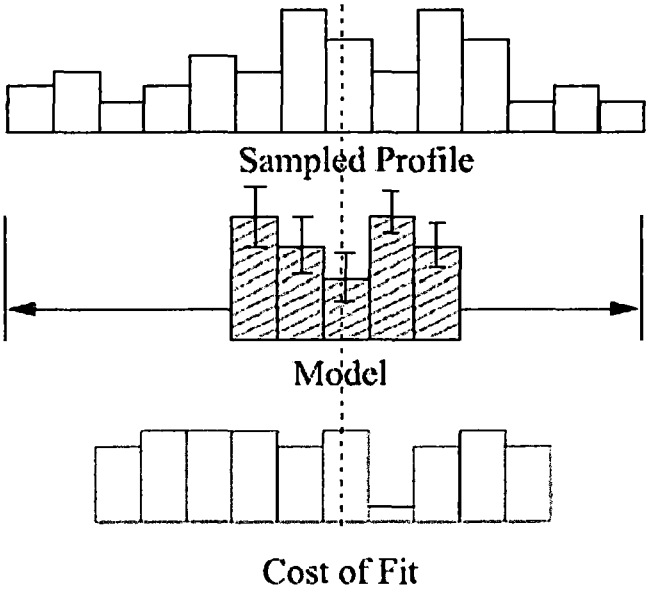
$$f(\mathbf{g}_s) = (\mathbf{g}_s - \bar{\mathbf{g}})^{\mathrm{T}}\mathbf{S}_g^{-1}(\mathbf{g}_s - \bar{\mathbf{g}}),$$

Y está relacionada al logaritmo negativo de la probabilidad que gs traza desde la distribución aprendida minimizando $f(\mathbf{g}_s)$ es equivalente a encontrar la solución de máxima verosimilitud.

En la práctica cuando se realiza una búsqueda local para un punto característico dado primero se muestrea un perfil de $m > k$ pixeles

a cualquier lado del estimado actual. Entonces probar la calidad de ajuste del correspondiente nivel de gris del modelo para cada uno de los $2(m - k) + 1$ posibles posiciones a lo largo de la muestra y escoger al que da la mejor coincidencia como muestra la FIGURA 2 que es, el menor valor de $f(g_s)$. Repitiendo esto para cada punto característico da un nuevo estimado para la forma del rostro.

FIGURA 2
BÚSQUEDA A LO LARGO DEL PERFIL MUESTREADO PARA ENCONTRAR EL MEJOR AJUSTE DE NIVEL DE GRIS DEL MODELO



Los gradientes de perfil han probado ser efectivos para búsquedas locales, los modelos discriminativos de perfiles de intensidad pueden distinguir entre coincidencias correctas e incorrectas y mejorar aún más el rendimiento (Van Ginneken, Frangi, Stall, & Ter Haar Romeny, 2002). Mejor aun usando parches en 2D en lugar de perfiles en 1D hacen el modelo aún

más discriminativos 44, basados sobre medidas como la correlación normalizada 21, la clasificación aumentada 20 o mixta de expertos lineales 50 para definir el puntaje de las coincidencias, donde observar el potencial de las coincidencias es usualmente definido manualmente (por ejemplo una grilla rectangular o elíptica) o también aprendido de los datos de entrenamiento 38.

2.2.2. MODELO ACTIVO DE APARIENCIA (ACTIVE APAREANCE MODEL - AAM)

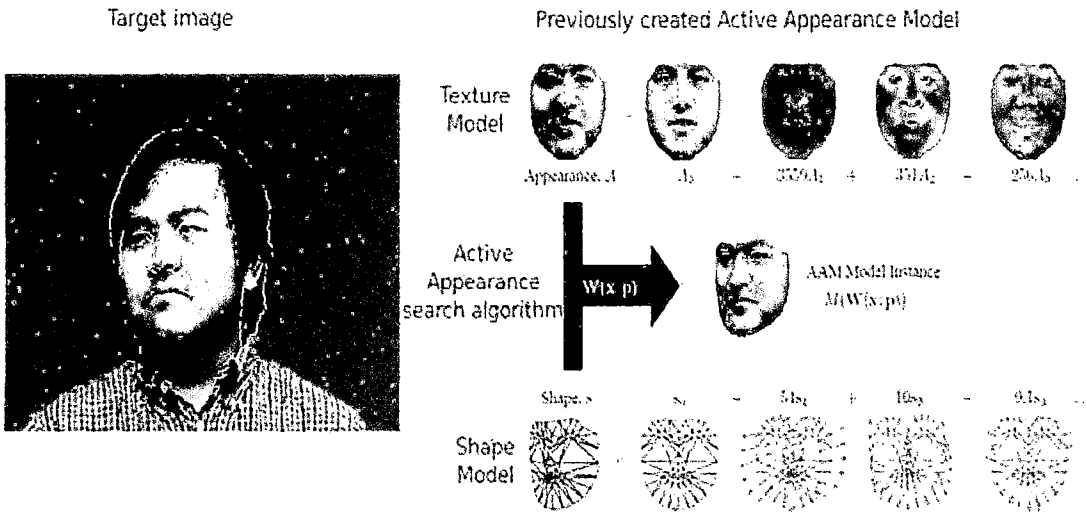
Un punto crítico de la aproximación realizada por ASM es que usa solo escasa información local alrededor de los puntos de interés. Adicionalmente ellos tratan la información en cada punto de manera independiente. Estos puntos críticos son tratados por los modelos de apariencia activa AAM.

En pocas palabras, los Modelos de Apariencia Activa son modelos parametrizados que combinan textura y forma acopladas a un eficiente algoritmo de búsqueda que pueden decir exactamente dónde y cómo un modelo está localizado en un marco de imagen.

La idea principal es deformar un modelo de formas, únicamente en direcciones provenientes de una etapa de entrenamiento, asegurando de esta manera que los modelos generados sean expresiones reales. Quitando así la posibilidad de que algunos puntos desaparezcan o que los puntos se desplacen de una manera no coherente respecto del resto del modelo.

Los Modelos Activos de Apariencia son un método estadístico de ajuste entre un modelo que representa forma y apariencia, y una imagen de entrada dada. En los AAM las variaciones de forma y de textura son capturadas a partir de un conjunto representativo de entrenamiento. Se busca entonces realizar entrenamientos que abarquen todas las transformaciones posibles que el modelo tiene que ser capaz de reproducir, a la hora del ajuste con la imagen. La FIGURA 3 resume el procedimiento general para un modelo de apariencia activa.

FIGURA 3
PROCESO PARA LA APLICACIÓN DE UN AAM



2.2.3. ANALISIS DE COMPONENTES PRINCIPALES (PRINCIPAL COMPONENT ANALYSIS - PCA)

El Análisis de Componentes Principales (Principal Component Analisys - PCA) es una técnica de reducción de dimensionalidad basado en la extracción de un número deseado de *componentes principales* de los datos multideimensinales. Los primeros

componentes principales son la combinación lineal de la dimensión original que tiene la máxima varianza; el enésimo componente principal es la combinación lineal con la varianza más alta, sujeto a ser ortogonales a los primeros n-1 componentes principales.

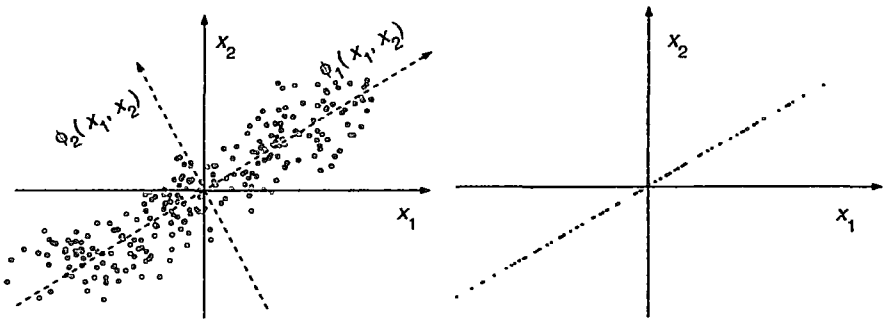
La idea de PCA está ilustrada en la FIGURA 4 a; los ejes con las etiquetas ϕ_1 corresponde a la dirección de la máxima varianza y es escogido como el primer componente principal. En un caso de dos dimensiones, el segundo componente principal esta entonces determinado únicamente por las restricciones de ortogonalidad; en un espacio de alta dimensionalidad el proceso de selección continuaría, guiado por las varianzas de las proyecciones.

El PCA está cercanamente relacionado a la transformada Karhunen.Loeve (KLT) (Loeve, 1955), Que fue derivado en el contexto de procesamiento de señales como la trasformada ortogonal con las bases $\phi = [\phi_1, \ldots, \phi_N]^T$ que para cualquier $k \leq N$ minimiza el error de reconstrucción promedio L_2 para los puntos de datos x.

$$\varepsilon(x) = \left\| x - \sum_{i=1}^{k} (\phi_i^{\mathrm{T}} x)\phi_i \right\|.$$

FIGURA 4

EL CONCEPTO DE PCA/KLT, A LINEAS SOLIDAS, LAS BASES ORIGINALES; LÍNEAS PUNTEADAS, LAS BASES KLT. LOS PUNTOS SON SELECCIONADOS REGULARMENTE EN UBICACIONES ESPACIADAS SOBRE UNA LÍNEA RECTA ROTADA EN 30° Y ENTONCES PERTURBADA POR UN RUIDO GAUSEANO ISOTRÓPICO EN 2D. B LA PROYECCIÓN (ID RECONSTRUCCIÓN) DE LOS DATOS USANDO SOLO EL PRIMER COMPONENTE PRINCIPAL



(a) PCA basis (b) PCA reduction to 1D

Uno puede mostrar (Gerbrands, 1981) que, bajo la presunción que los datos tienen media cero, la formulación de PCA y KLT son idénticas. Sin pérdida de generalidad, a continuación asume que los datos son en efecto de media cero; esto es, el rostro promedio $\bar{x}$ esta siempre derivada de los datos.

Los vectores base en KLT pueden ser calculadas de la siguiente manera. Sea X la matriz de datos de N x M con columnas $x_1, \ldots, x_M$, que son observaciones de una señal que pertenece a $\mathbb{R}^N$; en el contexto de reconocimiento de rostros, M es el número de imágenes de rostros disponible, y N = m n es el número de pixeles en una imagen. Las bases KLT ϕ es obtenida de resolver el problema de la obtención de los valores propios (eigenvalue) $\Lambda = \phi^T \Sigma \phi$, donde Σ es la matriz de covarianza de los datos

$$\Sigma = \frac{1}{M}\sum_{i=1}^{M} x_i x_i^{\mathrm{T}}$$

ϕ= ⟦ ⟦[ϕ]⟧ _1,...,ϕ_m]⟧ ^T es el eigenvector de Σ, y Λ es la matriz diagonal con eigenvalues λ_1≥⋯ ≥ λ_N de Σ sobre su diagonal principal, asíϕ_j es el eigenvector correspondiente al j-taesimo eigenvalue más grande. Entonces puede estar mostrando que los eigenvalues λ_i es la variancia de los datos proyectados sobre ϕ_i.

Así, para realizar PCA y extraer los k componentes principales de los datos, se debe proyectar los datos sobre ϕ_k, las primeras k columnas de las bases KLT ϕ, la cual corresponde a los k eigenvalues más altos de Σ Esto puede ser visto como una proyección lineal R^(N→) R^k, la cual retiene la máxima energía (por ejemplo variancia) de la señal. Otra propiedad importante de PCA es que decorrelaciona los datos: la matriz de covarianza de ϕ(_K^T)X es siempre diagonal.

Las principales propiedades de PCA son resumidos a continuación

$$x \approx \Phi_k y, \qquad \Phi_k^{\mathrm{T}}\Phi_k = I, \qquad E\{y_i y_j\}_{i\neq j} = 0$$

A saber, reconstrucción aproximada, ortonormalidad de las bases ϕ_k, y componentes principales decorrelacionados $y_i = \phi_i^T x$, respectivamente, Estas propiedades son ilustradas en la FIGURA 12, donde PCA encuentra de forma satisfactoria los colectores

principales, y en la FIGURA 13, donde es menos satisfactorio, debido a la clara no linealidad de los principales colectores.

El PCA puede ser implementado via la descomposición de valor singular (SVD). LA SVD de una matriz X de dimensión M x N (M >= N) está dada por

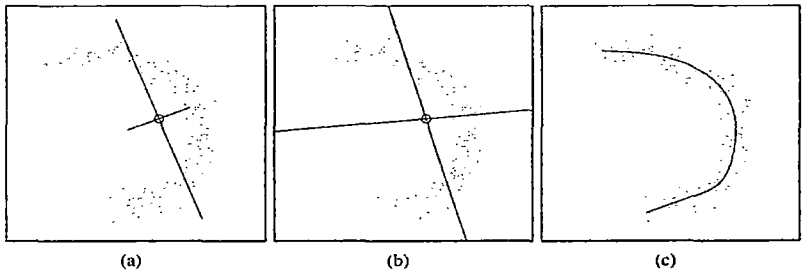
$$X = UDV^{\mathrm{T}} \qquad (2.47)$$

Donde la matriz $U\,de\,M \times N$ y la matriz $V\,de\,N \times N$ tiene columnas ortonormales y la matriz $\mathrm{D}\,de\,N \times N$ tiene los valores singulares de X sobre su diagonal principal y cero en otro caso.

Se puede demostár que $U = \phi$, asi SVD permite un cálculo eficiente y robustode PCA sin la necesidad de estimar la matriz de covarianza Σ. Cuando el número de muestras M es más pequeño que la dimensión N, esto es una ventaja crucial.

FIGURA 5

(a) BASES DEL PCA (LINEAL, ORDENADO, ORTOGONAL). (b) BASES ICA Y (c) CURVA PRINCIPAL.



2.2.4. TRIANGULACIÓN DE DELAUNAY (DT)

Es una red de triángulos que cumple la condición de Delaunay. Esta condición dice que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo. Se usan triangulaciones de Delaunay en geometría por ordenador.

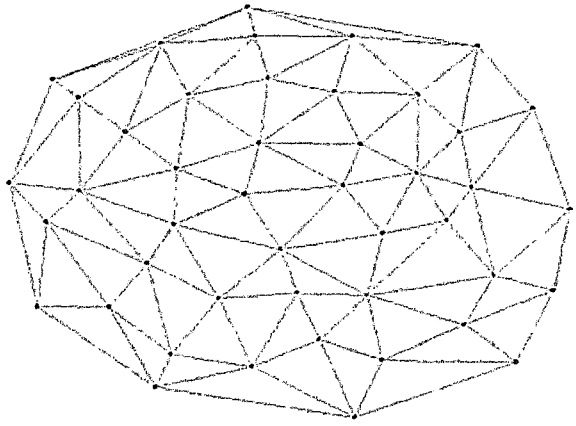
La circunferencia circunscrita de un triángulo es la circunferencia que contiene los tres vértices del triángulo. Según la definición de Delaunay la circunferencia circunscrita es vacía, si no contiene otros vértices aparte de los tres que la definen.

La condición de Delaunay dice que una red de triángulos es una triangulación de Delaunay si todas las circunferencias circunscritas de todos los triángulos de la red son vacías. Esa es la definición original para espacios bidimensionales. Es posible ampliarla para espacios tridimensionales usando la esfera circunscrita en vez de la circunferencia circunscrita. También es posible ampliarla para espacios con más dimensiones pero no se usa en la práctica.

Esa condición asegura que los ángulos del interior de los triángulos son lo más grandes posible. Es decir, maximiza la extensión del ángulo más pequeño de la red optimizando el número total generado (Natividad Grandon, 2007), podemos ver una descripción gráfica en la FIGURA 6.

FIGURA 6

EJEMPLO DE GENERACIÓN DE MALLAS POR TRIANGULACIÓN DE DELAUNAY



Propiedades

La triangulación forma la envolvente convexa del conjunto de puntos.

El ángulo mínimo dentro de todos los triángulos está maximizado.

La triangulación es unívoca si en ningún borde de circunferencia circunscrita hay más que tres vértices.

Relación con los diagramas de voronoi

La triangulación de Delaunay con todos los circuncentros es el grafo dual del diagrama de Voronoi: los circuncentros son los vértices de los segmentos del diagrama:

FIGURA 7

LA TRIANGULACIÓN CON TODAS LAS CIRCUNFERENCIAS CIRCUNSCRITAS Y SUS CENTROS.

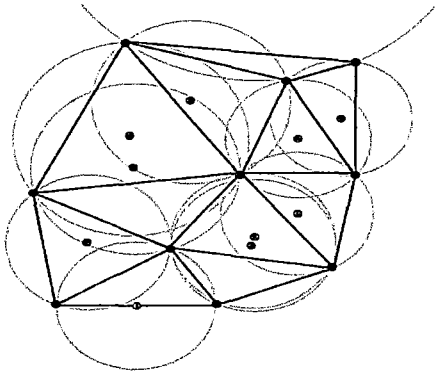


FIGURA 8

CONECTANDO LOS CENTROS DE LAS CIRCUNFERENCIAS CIRCUNSCRITAS SE PRODUCE EL DIAGRAMA DE VORONOI.

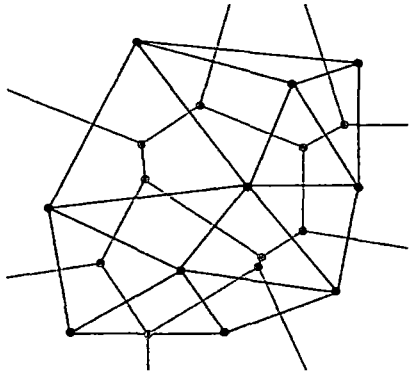
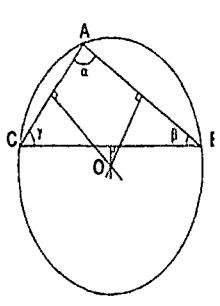
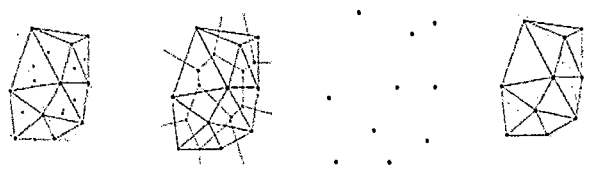


FIGURA 9

VÉRTICES A, B, C DEL TRIÁNGULO ABC MISMA DISTANCIA AL CIRCUNCENTRO O





$$\begin{vmatrix} A_x & A_y & A_x^2+A_y^2 & 1 \\ B_x & B_y & B_x^2+B_y^2 & 1 \\ C_x & C_y & C_x^2+C_y^2 & 1 \\ D_x & D_y & D_x^2+D_y^2 & 1 \end{vmatrix} = \begin{vmatrix} A_x-D_x & A_y-D_y & (A_x-D_x)^2+(A_y-D_y)^2 \\ B_x-D_x & B_y-D_y & (B_x-D_x)^2+(B_y-D_y)^2 \\ C_x-D_x & C_y-D_y & (C_x-D_x)^2+(C_y-D_y)^2 \end{vmatrix} > 0$$

2.2.5. TRANSFORMACIÓN AFIN

Las transformaciones Geométricas modifican la relación espacial entre píxeles. En términos del procesamiento de imágenes digitales una transformación geométrica consiste de dos operaciones básicas:

- Una transformación espacial que define la reubicación de los píxeles en el plano imagen.
- Interpolación de los niveles de grises, los cuales tienen que ver con la asignación de los valores de intensidad de los píxeles en la imagen transformada.

En términos Matemáticos las transformaciones afines son las más usadas en imágenes digitales 2D por su representación y manejo matricial.

Una Transformación afín es aquella (transformación) en la que las coordenadas (X',Y') del punto imagen son expresadas linealmente en términos de las del punto original *(X,Y)*. Es decir, la transformación viene dada por las ecuaciones:

$$x' = ax + by + m$$
$$y' = cx + dy + n$$

Cuando $m = n = 0$ las ecuaciones anteriores se convierten en el prototipo de una transformación lineal multiplicación por una matriz a la izquierda, esto es,

$$\begin{cases} x' = ax + by \\ y' = cx + dy \end{cases} \longrightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

El sistema tiene solución única si y sólo si

$$\Delta = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0$$

, es decir, la matriz es no singular, por lo tanto la transformación inversa existe y viene dada por:

$$\begin{cases} x = a'x' + b'y' \\ y = c'x' + d'y' \end{cases} \longrightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

La imagen de una recta es una recta:

$$\begin{aligned} Ax + By + D = 0 &\longrightarrow A(a'x' + b'y') + B(c'x' + d'y') + D = 0 \\ &\longrightarrow (Aa' + Bc')x' + (Ab' + Bd')y' + D = 0 \\ &\longrightarrow A'x' + B'y' + D = 0 \end{aligned}$$

Por lo anterior, no es difícil mostrar que las transformaciones afines transforman una grilla(o malla) en otra grilla, caracterizándolas como las más naturales a las imágenes.

En el estudio del significado de los coeficientes de la matriz de la transformación se puede comprobar que sí

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} SC_x & S_{Horiz.} \\ S_{Vert.} & SC_y \end{pmatrix}$$

Donde, $a = SC_x$ corresponde al escalado en *x*, $d = SC_y$ escalado en *y*, $b = S_{horiz}$ presión o inclinación en dirección vertical.

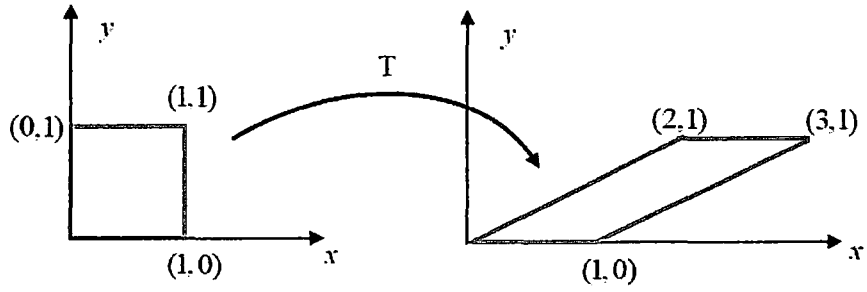
Ejemplo: sea

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + 2y \\ y \end{pmatrix}.$$

El efecto de los coeficientes de la matriz e la transfomación puede observarse en la FIGURA 10.

FIGURA 10

TRANSFORMACIÓN PRESIÓN O EMPUJE EN DIRECCIÓN HORIZONTAL



Existen también otras acciones de las transformaciones afines tal como la de rotar una región geométrica un ángulo alrededor de un punto fijo y que se define como:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Un punto (*x, y*) en coordenadas homogéneas se representa por la terna (*xw, yw, w*). Es así, como el punto (3, 2) del sistema de coordenadas cartesianas se representa como (6, 4, 2), o como (15, 10, 5) entre otros. Y la representación normalizada de (3, 2) en coordenadas homogéneas es (3, 2, 1) que es la convención más usada.

La representación matricial en coordenadas homogéneas de la transformación afín 2D:

$$\begin{cases} x' = ax + by + m \\ y' = cx + dy + n \end{cases} \longleftrightarrow \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & m \\ c & d & n \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Donde los escalares a, b, c, d tienen el mismo significado que los de la ecuación (4),m corresponde al desplazamiento horizontal mientras n al desplazamiento vertical. En la FIGURA 2 se ilustran la aplicación de una transformación afín a una imagen cuya acción

es la de una presión o empuje en dirección horizontal y luego una acción compuesta de rotación, escalado y traslación.

FIGURA 11

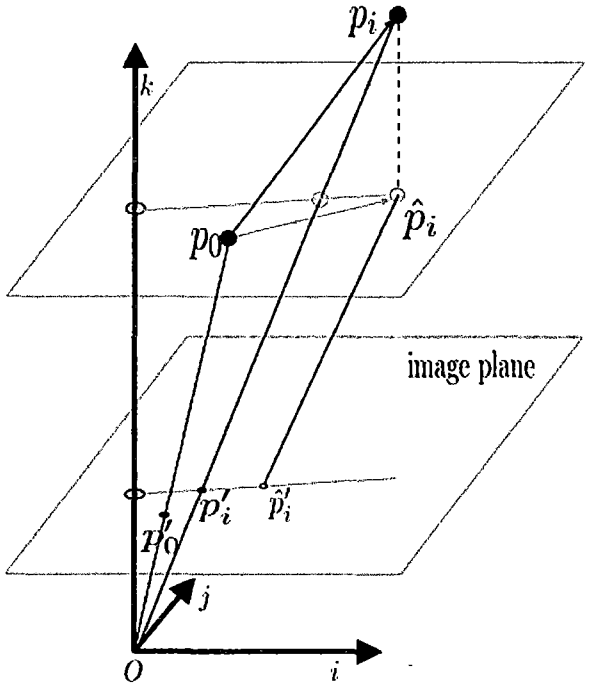
TRANSFORMACIÓN AFIN ACTUANDO SOBRE UNA IMAGEN DIGIRAL

2.2.6. POSE FROM ORTHOGRAPHY AND SCALING WITH ITERATION – POSIT

POSIT usa una proyección ortográfica escalada (Scaled orthographic projection - SOP), la que se asemeja a la real proyección de la perspectiva en convergencia. La aproximación de SOP trata con un sistema de ecuaciones lineales.que da la rotación y traslación directamente, sin la necesidad de una postura inicial. Un valor de escala es introducido para cada correspondencia, la que es actualizada de forma iterativa. La FIGURA muestra lo mencionado.

FIGURA 12

POSIT ESTIMACIONES DE LA POSTURA POR USAR UNA PROYECCIÓN ESCALADA ORTOGRÁFICA (SOP) DADAS LAS CORRESPONDENCIAS PI, PI'. LA SOP DE PI ES AQUÍ MOSTRADO COMO P"I CON UN VALOR DE ESCALA DE 0.5.



Las correspondencias son p_i, p_i'. La SOP de p_i es mostrada como p_i' con un valor de escala de 0.5. El algoritmo de POSIT estima la rotación encontrando los valores de ***i, j, k*** en el sistema de coordenadas del objeto, cuyo origen es p_0. La traslación entre el objeto y el sistema de la cámara es $\boldsymbol{Op_0}$.

Para cada punto SOP 2D un valor escala puede ser encontrado de tal manera que la SOP $\hat{p}'_i$ iguala la correcta proyección perspectiva p'_i.

El algoritmo POSIT refina iterativamente estos valores de escala. Inicialmente el valor de escala (*w* en lo siguiente) eta ajustado a uno.

El algoritmo POSIT trabaja como sigue:

1. Inicializa el conjunto de valores desconocidos $w_i = 1$ para cada correspondencia.
2. Estima los parámetros de posición desde el sistema lineal de ecuaciones
3. Estima nuevos valores $w_i = \frac{p_i^T k}{t_z} + 1$
4. Repetir desde el paso 2 hasta el cambio en w_i este por debajo de una frontera o la máxima iteración es rechazada.

El $w_i = 1$ inicialmente escogido aproxima a la real conFIGURAción de la posición de la cámara y tambien a los puntos en a escena, Si la fracción de la expansión del objeto a la cámara es pequeña.

Si los puntos 3D caen un plano el algoritmo POSIT necesita ser modificado.

2.2.7. MODELO DE CALIDAD ISO/IEC 9126

ISO- 9126 es un Estándar Internacional para la evaluación del software y tiene como objetivo la definición de un modelo de calidad y su uso como marco para la evaluación de software. Los modelos de calidad concordantes con este estándar pertenecen a la categoría de modelos mixtos, ya que el estándar propone una jerarquía de factores de calidad clasificados como características, sub características y atributos según su grado de abstracción, entre los que se propone un conjunto de factores de partida

compuestos de 6 características y 27 sub características, las cuales se describen a continuación:

a) Funcionalidad.- La capacidad del producto software para proporcionar funciones declaradas e implícitas cuando se usa bajo condiciones especificadas:

- Adecuación.
- Exactitud.
- Interoperabilidad.
- Seguridad de acceso.
- Cumplimiento funcional.

b) Fiabilidad.- La capacidad del producto software para mantener un nivel especificado de prestaciones cuando se usa bajo condiciones especificadas:

- Madurez.
- Tolerancia a fallos.
- Capacidad de recuperación.
- Cumplimiento de la fiabilidad.

c) Usabilidad.- La capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones especificadas:

- Capacidad para ser entendido.
- Capacidad para ser aprendido.
- Capacidad para ser operado.
- Capacidad de atracción.
- Cumplimiento de la usabilidad.

d) Eficiencia.- La capacidad del producto software para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas:

- Comportamiento temporal.

- Utilización de recursos.
- Cumplimiento de la eficiencia.

e) Mantenibilidad.- La capacidad del producto software para ser modificado. Las modificaciones podrían incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y requisitos y especificaciones funcionales:

- Capacidad para ser analizado.
- Capacidad para ser cambiado.
- Estabilidad.
- Capacidad para ser probado.
- Cumplimiento de la mantenibilidad.

f) Portabilidad.- La capacidad del producto del software para ser transferido de un entorno a otro, puede incluir la siguientes especificaciones funcionales:

- Adaptabilidad.
- Instalabilidad.
- Coexistencia.
- Capacidad para reemplazar.
- Cumplimiento de la portabilidad.

2.3. MARCO CONCEPTUAL

2.3.1. MODELO ESTADISTICO

Un modelo estadístico es una expresión simbólica en forma de igualdad o ecuación que se emplea en todos los diseños experimentales y en la regresión para indicar los diferentes factores que modifican la variable de respuesta. Los modelos estadísticos pueden ser lineales o no lineales.

2.3.2. LAND MARK

Un Land Mark en inglés y puntos de referencia en español son los puntos ubicados dentro del marco de una imagen. Estos puntos indican la ubicación de puntos que contienen información relevante del objeto, la ubicación correcta de un land mark en la imagen es fundamental para un mejor rendimiento en la tarea de matching.

2.3.3. IMAGEN DIGITAL

Una imagen digital es una imagen f(x,y) que se ha discretizado tanto en las coordenadas espaciales como en el brillo. Una imagen digital puede considerarse como un matriz cuyos índices de fila y columna identifican un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de brillo en ese punto (Gonzales, 2007).

2.3.4. PIXEL

Los elementos de una distribución digital (imagen digital) de este tipo se denominan elementos de la imagen, o más comúnmente

pixeles o pels abreviaturas de su denominación inglesa picture elements (Gonzales, 2007).

2.3.5. MATCHING

Matching es el acto de comprobar una secuencia de ítems para determinar la presencia de algún patrón. En contraste al reconocimiento de patrones, el match usualmente tiende a ser exacto. Los patrones generalmente tienen la forma tanto de secuencias como estructuras de árbol. El uso de matching incluye la respuesta de ubicación de un patrón dentro de una secuencia determinada y el reemplazo de alguna otra secuencia. Los patrones de árbol son herramientas generales para procesar datos basados en su estructura.

2.3.6. ANALISIS PROCRUSTER

Es un tipo de análisis estadístico de forma utilizado para analizar la distribución de un conjunto de formas. La superimposición Procrustes es realizada óptimamente trasladando, rotando y escalando uniformemente los objetos.

2.3.7. CALIDAD

Grado en que un conjunto de características inherentes cumple con los requisitos.

2.3.8. CALIDAD INTERNA

Es la totalidad de características del producto que determinan su habilidad para satisfacer las necesidades establecidas e implícitas bajo condiciones específicas.

2.3.9. CALIDAD EXTERNA

Es la totalidad de características que determina hasta qué punto un producto satisface las necesidades explicitas e implícitas cuando es usado bajo condiciones específicas.

2.3.10. ISO/IEC 9126

Es un estándar internacional para la evaluación de la calidad del software.

2.3.11. MÉTRICA

Es un método definido de valoración y su escala de valoración.

2.3.12. PRODUCTO SOFTWARE

Es un producto del proceso de desarrollo del software que se emplea para alimentar una etapa diferente del proceso de desarrollo

CAPÍTULO III

METODOLOGÍA

3.1. TIPO DE INVESTIGACIÓN

La presente investigación corresponde al tipo de investigación experimental donde se manipularon las técnicas para lograr los propósitos propuestos.

3.2. DISEÑO DE INVESTIGACIÓN

El diseño de investigación corresponde al pre-experimental.

GXO

Donde:

G: Grupo Experimental imágenes de entrenamiento.

X: Tratamiento: software de AAM y POSIT

O: Instrumento: Métricas de calidad interna y calidad externa.

3.3. POBLACION Y MUESTRA

La población estudiada estuvo conformada por todas las métricas de calidad de la norma ISO/IEC 9126.

CUADRO 1

POBLACION DE MÉTRICAS PARA EL SOFTWARE

Métricas de calidad	Frecuencia Absoluta	Frecuencia Relativa	Frecuencia Porcentual
Métricas internas	70	0.40	40%
Métricas Externas	105	0.60	60%
TOTAL	175	1.00	100%

Fuente: Elaboración propia

La selección de la muestra es del tipo no probabilístico, se utilizó el muestreo por conveniencia debido a la cualidad de obtener las muestras accesibles representativas. Por tanto se tomó como muestra a las métricas más significativas y pertinentes al software desarrollado.

CUADRO 2

MUESTRA DE MÉTRICAS PARA EL SOFTWARE

Métricas de calidad	Frecuencia Absoluta	Frecuencia Relativa	Frecuencia Porcentual
Métricas internas	5	0.42	42%
Métricas Externas	7	0.58	58%
TOTAL	12	1.00	100%

Fuente: Elaboración propia

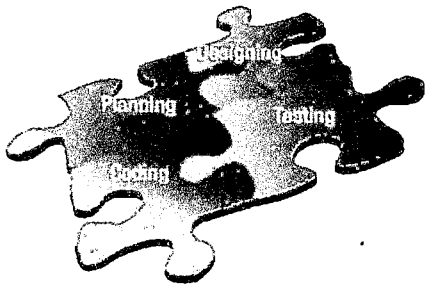
3.4. METODOLOGÍA DE DESARROLLO

3.4.1. PROGRAMACIÓN EXTREMA

Para el desarrollo del Software de compresión se aplicó la metodología de la Programación Extrema (Extremme Programming - XP) que se adapta hoy en día perfectamente al desarrollo del ciclo de vida del Software y para el modelado del Software se usó el Lenguaje Unificado de Modelamiento (Unified Modeling Languaje - UML).

FIGURA 13

FASES DE LA METODOLOGÍA XP



Las fases de la programación extrema se dividen en cuatro fases:

1. Análisis: La metodología XP plantea en análisis como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán el alcance ¿Qué es lo realmente necesario del proyecto?, la prioridad qué debe ser hecho en primer lugar, la composición de las versiones que debería incluir cada una de ellas y la flecha de las mismas en cuanto a los técnicos, son los responsables de estimar la duración requerida para implementar las funcionalidades deseadas por el cliente, de informar sobre las consecuencias de determinadas decisiones, de organizar la cultura de trabajo y finalmente de realizar la planificación detallada dentro de cada versión. XP no solo es un método centrado en el código que lo es, sino que sobre todo es un método de gestión de proyectos software (Grady Booch, 1999).

2. Diseño: El Propósito del diseño es de crear una arquitectura para la naciente implementación, el diseño arquitectural sólo puede comenzar una vez que el equipo tenga un entendimiento

razonable de los requerimientos del sistema. El diseño, como el análisis, nunca termina realmente hasta que el sistema final es entregado. Durante esta fase se alcanza un cierto diseño y al establecer políticas para diversos problemas tácticos.

El diseño se enfoca en la estructura, estática y dinámica, su propósito principal es de crear el esqueleto concreto del sistema sobre el cual todo el resto de la implementación se basa (Grady Booch, 1999)

3. Desarrollo: Esta etapa debe reunir las siguientes características o cualidades :

- El software está siempre disponible
- Se debe escribir código de acuerdo a los estándares
- Desarrollar la unidad de pruebas primero
- Todo el código debe programarse por parejas
- Integrar frecuentemente
- Todo el código es común a todos

Prueba: Todo el código debe ir acompañando, Los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.

4. Evaluación. Esta etapa es realizada aplicando la norma ISO/IEC 9126 donde detalla el modelo a usar para la calidad del producto software, tanto la calidad interna como la calidad externa. El estándar ISO/IEC 9126 puede ser usado desde varias

perspectivas, en esta investigación será usado de la perspectiva de desarrollo.

CAPÍTULO IV

RESULTADOS

En el presente capítulo se expone los resultados organizados primero por la identificación de los requisitos, análisis y diseño del software, implementación del software y la evaluación de la efectividad del software.

4.1. REQUISITOS

Los requisitos de software se registraron mediante las historias de los usuarios y las historias de los desarrolladores.

4.1.1. HISTORIAS DE USUARIO

Numero: 01	**Historias de Usuario**
Nombre Historia	Iniciar la Aplicación.
Quiero: Iniciar la aplicación de forma local sin usar usuario y contraseña.	
Para: Poder acceder a las funciones de la aplicación de procesamiento, detección y reconocimiento de rostro.	

Numero: 02	**Historias de Usuario**
Nombre Historia	Captura de Imagen.
Quiero: Imagen desde el programa.	
Para: almacenar imágenes y poderlas analizar.	

Numero: 03	**Historias de Usuario**
Nombre Historia	Captura de Video.
Quiero: Capturar segmentos de video desde el programa.	
Para: Almacenar segmentos de videos y poderlas analizar.	

Numero: 04	**Historias de Usuario**
Nombre Historia	Selección de imagen/video desde archivo
Quiero: seleccionar imagen/video desde archivo	
Para: poderlas analizar desde el programa.	

Numero: 05	Historias de Usuario
Nombre Historia	Detección de Rostros en imágenes/videos
Quiero: Que el programa seleccione los rostros que existen en una imagen.	
Para: Poder trabajar solamente con los rostros de las personas que existan dentro de las imágenes omitiendo información no relevante.	

Numero: 06	Historias de Usuario
Nombre Historia	Diferenciar rostros de otros objetos en las imágenes
Quiero: Que el programa seleccione solamente las imágenes de rostros.	
Para: Almacenar las imágenes de rostros en la base datos para posteriormente analizarlas.	

Numero: 07	Historias de Usuario
Nombre Historia	Realizar marcas en la imagen - LAND MARKS
Quiero: Que el programa seleccione los puntos referenciales de las imágenes de rostros.	
Para: Almacenar la información de los puntos seleccionados de los rostros en la base datos para posteriormente analizarlas.	

Numero: 08	Historias de Usuario
Nombre Historia	Utilizar los land mark para generar ASM y AAM
Quiero: Que el programa utilice los puntos referenciales de las imágenes de rostros almacenados	
Para: generara con estos los modelos de forma y apariencia.	

Numero: 09	Historias de Usuario
Nombre Historia	Generar malla de triángulos utilizando una función de delaunay
Quiero: el programa genere una malla de triángulos.	
Para: proyectar sobre estas formas y texturas.	

Numero: 10	Historias de Usuario
Nombre Historia	Proyección de los modelos de formas y apariencias utilizando Análisis de componentes principales en los triángulos generados
Quiero: el programa proyecte la forma y apariencia utilizando PCA.	
Para: emparejar los modelos con las imágenes.	

Numero: 11	Historias de Usuario
Nombre Historia	Generar el modelo 3D en función del modelo 2D utilizandoel algoritmo POSIT.
Quiero: el programa proyecte la forma y apariencia utilizando PCA.	
Para: emparejar los modelos con las imágenes.	

4.1.2. TAREAS DE USUARIO

	Tarea de Ingeniería
Número Tarea: 01	**Número Historia: 01**
Nombre Tarea: Ejecutar la aplicación	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador: Programador 1	**Iteración:** 3
Descripción: El acceso a la aplicación se ejecuta mediante el administrador de aplicaciones del Sistema Operativo (El usuario lanzara la aplicación con doble clic, enter o mediante el Shell del sistema).	

Tarea de Ingeniería	
Número Tarea: 02	**Número Historia: 02, 03**
Nombre Tarea: Crear menús e iconos de acceso	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador: Programador 1	**Iteración:** 3
Descripción: El usuario podar acceder a las funciones de captura de pantalla a través del menú principal de la aplicación y mediante la barra de herramientas de la aplicación.	

Tarea de Ingeniería	
Número Tarea: 03	**Número Historia: 02, 03 y 04**
Nombre Tarea: Cargar archivos (imagen/video) desde archivo al programa.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador: Programador 1	**Iteración:** 3
Descripción: - El usuario inicia la carga del archivo desde el menú principal o la barra de herramientas de aplicación la imagen se carga en una ventana auxiliar donde se proyecta la imagen y/o video. - Se crean funciones de Carga de archivo en memoria para operar con ella.	

Tarea de Ingeniería	
Número Tarea: 04	Numero Historia: 05 y 06
Nombre Tarea: Ejecutar la detección de rostr os en imágenes/video.	
Tarea Asociada: Tarea 05, 03	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Lanza la función de apertura de archivo de imagen/video para ejecutar sobre esta la detección de rostros.	

Tarea de Ingeniería	
Nùmero Tarea: 05	Numero Historia: 05, 06
Nombre Tarea: Detección de Rostros.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:
El desarrollo de la detección de rostros usa "Boosted Cascade of Simple Features", modelo machine learning. Donde el modelo en cascada es entrenada con imágenes donde hay rostros y donde no las hay. Sigue el siguiente procedimiento.

- Dada una imagen de muestra $(x_1, y_1), \ldots, (x_n, y_n)$ donde yi = 0,1 para muestras negativas y positivas respectivamente.
- Inicializar ponderaciones $w_{1,i} = \frac{1}{2m}; \frac{1}{2l}$ para yi = 0,1 respectivamente, donde m y l son el número de negativos y positivos.
- Para t = 1,..., T:
- Normalizar las ponderaciones,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

Asi que *wt* es una distribución de probabilidad.

- Para cada característica, j, entrenar un clasificador *hj* que eta restringida a usar una característica simple. El erro es evaluado con respecto a *wt*

$$\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|.$$

- Escoger el casificador, *ht*, con el error menor . ϵ_j
- Actualizar las ponderaciones: $w_{t+1,i} = w_{t,i}\beta_t^{1-\epsilon_i}$, donde ei = 0 si la muestra, es clasificada correctamente, ei = 1 de otra forma, y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- El clasificador fuerte final es:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Donde:

$$\alpha_t = \log \frac{1}{\beta_t}$$

Tarea de Ingeniería	
Numero Tarea: 06	Numero Historia: 05 y 06
Nombre Tarea: Almacenar los rostros detectados.	
Tarea Asociada: Tarea 05.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Almacenar los rostros detectados en el directorio imágenes, las imágenes son ahora solo los rostros recortados de la imagen original..	

	Tarea de Ingeniería
Número Tarea: 07	Numero Historia: 07
Nombre Tarea: Realizar marcas referenciales en las imágenes de rostros LandMark.	
Tarea Asociada: Tarea 03	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: - Habilitar el marcado de puntos referenciales sobre imágenes. - La elección se realiza con un click sobre la coordenada actual donde se encuentra el puntero del mouse. - almacenar la información en archivo y posteriormente utilizar esta. - La información almacenada consta de posición de las coordenadas, niveles de los colores - La información debe ser almacena da en una estructura de datos.	

	Tarea de Ingeniería
Número Tarea: 08	Numero Historia: 07
Nombre Tarea: Realizar el entrenamiento con las marcas registradas utilizando Analisis de componentes principales.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Leer el archivo de las marcas realizadas y utilizarlas en el entrenamiento de PCA	

	Tarea de Ingeniería
Número Tarea: 09	Numero Historia: 08
Nombre Tarea: calcular los eigen values y vectores..	
Tarea Asociada:	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Se calcula el Análisis de Componentes Principales y prosigue como: El conjunto de entrenamiento será $\Gamma_1, \Gamma_2, \Gamma_3, \ldots, \Gamma_M$. El promedio del conjunto es definido por la ecuación : $$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i$$ Se diferencia del promedio por el vector de la ecuación:	

$$\Phi_i = \Gamma_i - \Psi, i = 1, \ldots, M$$

A continuación calculamos la matriz de covarianza C, según la ecuación:

$$C = \sum_{i=1}^{N} \Phi_i \Phi_i^T = AA^T$$

La matriz C es N2 por N2, y determinar los N2 vectores y valores propios se vuelve una tarea intratable para el tamaño de una imagen típica. Necesitamos un método factible para encontrar los eigen vectores ecuación.

$$\begin{pmatrix} e_i = & Av_i \\ \lambda_i = & \mu_i \end{pmatrix}$$

De M eigen vectores ei, solo escogemos M1 que tiene el valor propio (eigen value) más alto. Cuanto más alto sea el valor propio, describe la característica más representativa de un particular eigenvector. Los eigen bajos eigen values pueden ser omitidos, porque ellos solo explican una pequeña parte de los rasgos característicos.

Tarea de Ingeniería	
Número Tarea: 10	Numero Historia: 08
Nombre Tarea: Proyectar la forma de las marcas utilizando PCA.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Una vez calculado el modelo en función de los puntos referenciales almacenados proyectar la forma en función de este modelo.	

Tarea de Ingeniería	
Número Tarea: 11	Numero Historia: 09
Nombre Tarea: Generar malla de triángulos utilizando el algoritmo de Delanuay.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Generar subdivisiones de triángulos que incluyan los puntos anotados Mapear de un triángulo a otro Poblar de puntos en las subdivisiones La triangulación de Delanuay es realizada con los siguientes pasos:	

Tarea de Ingeniería	
Número Tarea: 12	Numero Historia: 09
Nombre Tarea: Deformación de la textura en los triángulos.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Deformar un triángulo de una imagen original anotada a una distorsionada generado. Calcular la matriz de la transformación afín utilizando: $$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \texttt{mapMatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$	

Tarea de Ingeniería	
Número Tarea: 13	Numero Historia: 09
Nombre Tarea: Búsqueda y Ajuste del AAM.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Encontrar el conjunto de parámetros de p formas y λ apariencias que aproxime nuestro modelo a la imagen de entrada, mapeando los puntos sobre la apariencia base y calcular la diferencia, minimizando la siguiente función: $$\sum_{\mathbf{x}\in \mathbf{s}_0} \left[A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2$$ Donde $S0$ denota el conjunto de pixeles x igual a $(x, y)T$ que se encuentran dentro de la malla base del AAM, $A_0(x)$ es la malla de textura base $A_i(x)$ es la imagen de apariencia desde PCA, y $W(x;p)$ es la deformación que tomana los pixeles de la imagen de entrada de vuelta a la malla base. El calcula de esta minimización se realiza según las siguientes instrucciones:	

Pre-compute:

(3) Evaluate the gradient ∇A_0 of the template $A_0(\mathbf{x})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{0})$
(5) Compute the modified steepest descent images using Equation (41)
(6) Compute the Hessian matrix using modified steepest descent images

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
(2) Compute the error image $I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - A_0(\mathbf{x})$
(7) Compute dot product of modified steepest descent images with error image
(8) Compute $\Delta \mathbf{p}$ by multiplying by inverse Hessian
(9) Update the warp $\mathbf{W}(\mathbf{x}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta \mathbf{p})^{-1}$

Post-computation:

(10) Compute λ_i using Equation (40). [Optional step]

La actualización ocurre en términos de un paso compuesto como en el paso (9)
Las ecuaciones 40 y 41 de las instrucciones anteriores son:

$$\lambda_i = \sum_{\mathbf{x} \in s_0} A_i(\mathbf{x}) \cdot [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - A_0(\mathbf{x})]. \quad (40)$$

$$\mathrm{SD}_j(\mathbf{x}) = \nabla A_0 \frac{\partial \mathbf{W}}{\partial p_j} - \sum_{i=1}^{m} \left[\sum_{\mathbf{x} \in s_0} A_i(\mathbf{x}) \cdot \nabla A_0 \frac{\partial \mathbf{W}}{\partial p_j} \right] A_i(\mathbf{x}) \quad (41)$$

	Tarea de Ingeniería
Número Tarea: 14	Numero Historia: 09
Nombre Tarea: Búsqueda y Ajuste del AAM.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:

Encontrar el conjunto de parámetros de *p* formas y λ apariencias que aproxime nuestro modelo a la imagen de entrada, mapeando los puntos sobre la apariencia base y calcular la diferencia, minimizando la siguiente función:

$$\sum_{\mathbf{x} \in s_0} \left[A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p})) \right]^2$$

Donde *S0* denota el conjunto de pixeles *x* igual a *(x, y)T* que se encuentran dentro de la malla base del AAM, $A_0(x)$ es la malla de textura base $A_i(x)$ es la imagen de apariencia desde PCA, y $W(x;p)$ es la deformación que tomana los pixeles de la imagen de entrada de vuelta a la malla base. El calcula de esta minimización se realiza según las siguientes instrucciones:

Pre-compute:

(3) Evaluate the gradient ∇A_0 of the template $A_0(\mathbf{x})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{0})$
(5) Compute the modified steepest descent images using Equation (41)
(6) Compute the Hessian matrix using modified steepest descent images

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
(2) Compute the error image $I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - A_0(\mathbf{x})$
(7) Compute dot product of modified steepest descent images with error image
(8) Compute $\Delta\mathbf{p}$ by multiplying by inverse Hessian
(9) Update the warp $\mathbf{W}(\mathbf{x}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$

Post-computation:

(10) Compute λ_i using Equation (40). [Optional step]

La actualización ocurre en términos de un paso compuesto como en el paso (9)
Las ecuaciones 40 y 41 de las instrucciones anteriores son:

$$\lambda_i = \sum_{\mathbf{x} \in s_0} A_i(\mathbf{x}) \cdot [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - A_0(\mathbf{x})], \tag{40}$$

$$\mathrm{SD}_j(\mathbf{x}) = \nabla A_0 \frac{\partial \mathbf{W}}{\partial p_j} - \sum_{i=1}^{m} \left[\sum_{\mathbf{x} \in s_0} A_i(\mathbf{x}) \cdot \nabla A_0 \frac{\partial \mathbf{W}}{\partial p_j} \right] A_i(\mathbf{x}) \tag{41}$$

	Tarea de Ingeniería
Número Tarea: 15	Numero Historia: 09
Nombre Tarea: Derivar la posición 3D de nuestro modelo usando POSIT.	
Tarea Asociada: Tarea 09	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:

Paso 0. Escribir la matriz **A** con dimensión (N-1) x 3; cada vector fila es un vector M_0M_i conectando el punto característico de referencia M_0 a otro punto característico M_i; calcula la 3 x (N-1) matriz **B** como la pseudoinversa matriz de **A;**

Paso 1. $\varepsilon_{i(0)} = 0, \ (i = 1 \ldots N-1), \ n = 1;$

Paso 2. Inicializar el bucle:

Calcular **I, j** y Z_0 :

- Calcular el vector de imagen x' con N-1 coordenadas de la forma $x_i(1 + \varepsilon_{i(n-1)}) - x_0$, y el vector de imagen y' con N-1 coordenadas de la forma $y_i(1 + \varepsilon_{i(n-1)}) - y_0$,
- Multiplicar la matriz objeto **B** de 3x(N-1) y el vector de imagen (N-1 coordenadas) para obtener vectores **I** y **J** con 3 coordenadas: **I=Bx'** y **J=By';**

- Calcular la escala s de la proyección como el promedio entre la norma de **I** y **J** :

$$s_1 = (\mathbf{I} \cdot \mathbf{I})^{1/2}, \; s_2 = (\mathbf{J} \cdot \mathbf{J})^{1/2}, \; s = (s_1 + s_2)/2$$

- Calcular los vectores unitarios **i** y **j** :

$$\mathbf{i} = \mathbf{I}/s_1, \; \mathbf{j} = \mathbf{J}/s_2;$$

Paso 3. Calcular nuevo ε_i

- Calcular el vector unitario **k** como el producto cruzado de **i** y **j.**
- Cacular la z-coordenada Z_0 del vector de traslación como

$Z_0 = f/s$ donde f es la longitud focal de la cámara.

- Calcular $\frac{1}{Z_0}\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{k};$

Paso 4. Si $|\varepsilon_{i(n)} - \varepsilon_{i(n-1)}| > \text{Threshold}$ n = n+1; ir al Paso 2.

Paso 5. Si la postura respuesta usando valores es hallada en la ultima iteración: El vector de traslación completa $\mathbf{OM}_0$ is $\mathbf{OM}_0 = \mathbf{Om}_0/s;$ la matriz de rotación es la matriz con los vectores fila **i, j** y **k**; para aplicaciones donde la matriz de rotación podría ser perfectamente ortonormal. Renormalizar esta matriz: calcular **k' = k/|k|, j' = k' x i**, y la respuesta de la matriz con vector fila **i, j' y k'.**

4.2. ANÁLISIS DEL SISTEMA

4.2.1. DIAGRAMAS DE CASOS DE USO

En la FIGURA 14 se observa el diagrama de casos de uso que resumen las funciones del software, estas funciones están orientadas a la adquisición de recursos imágenes y video, generación del Modelo de Apariencia Activa (incluye la lectura de los land mark, generación del modelo con PCA, triangulación de delanuay, y proyección de los modelos de forma y de apariencia), Busqueda y ajuste del modelo más próximo al almacenado, y la generación del modelo tridimensional utilizando el algoritmo POSIT.

FIGURA 14.

DIAGRAMA DE CASOS DE USO DEL SOFTWARE

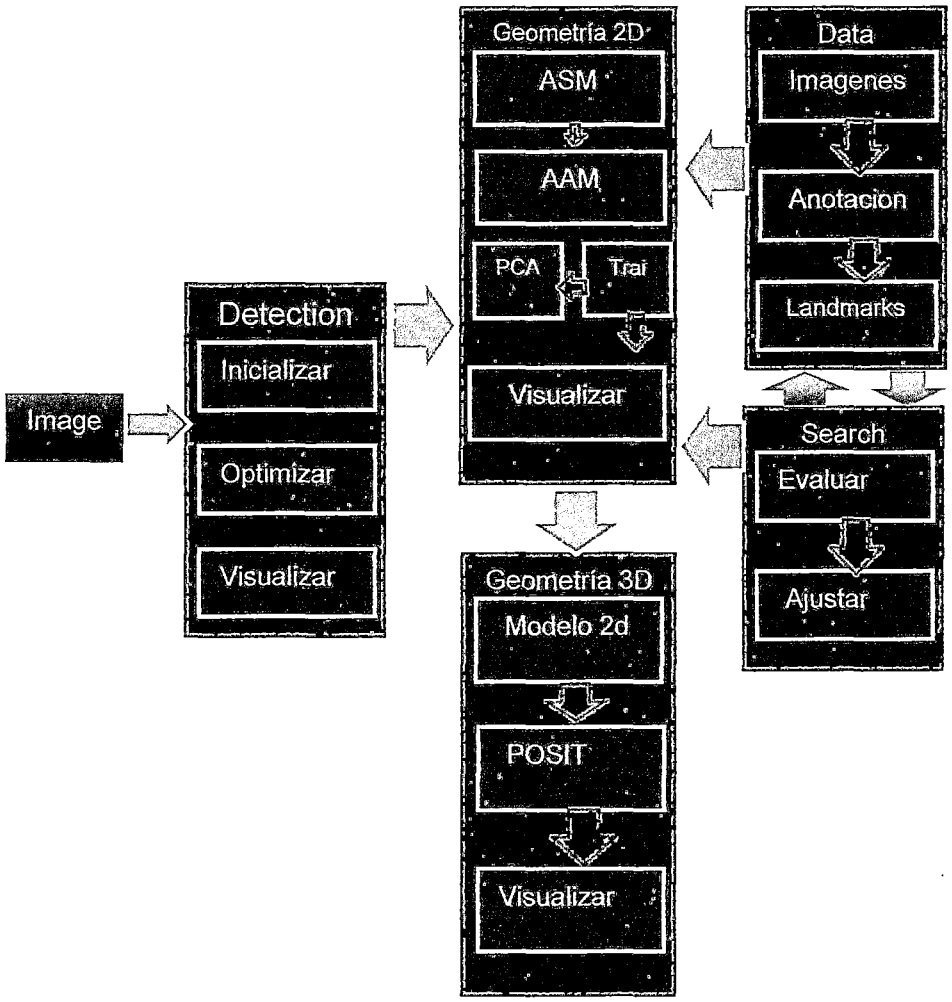
4.3. DISEÑO DEL SISTEMA

4.3.1. DISEÑO ARQUITECTÓNICO

La FIGURA 15 resume en un diagrama de bloques la arquitectura de los componentes principales del software. Los componentes encargados de la geometría en 2D están compuesto de los módulos ASM, AAM, PCA y la triangulación del modelo incluyendo los puntos referenciales. Los componentes de la geometría en 3D incluyen al modelo en 2D el algoritmo POSIT y la visualización.

FIGURA 15.

DIAGRAMA DE BLOQUES DE LA ARQUITECTURA DEL SOFTWARE.



Además el diagrama también muestra los componentes data, Búsqueda - ajuste y la detección automática de rostros. Se aprecia que la interacción con la data es realizada por los componentes de búsqueda y Geometría 2D como también la interacción que existe entre Geometría 2D y Busqueda y ajuste. Finalmente el flujo de la interacción de la detección, data y búsqueda va dirigida hacia el componente geometría 2D para finalmente dirigir el flujo al componente Geometría 3D que es la encargada de proyectar el modelo generado en 2D a un modelo en 3D.

4.3.2. DISEÑO DE MODULOS

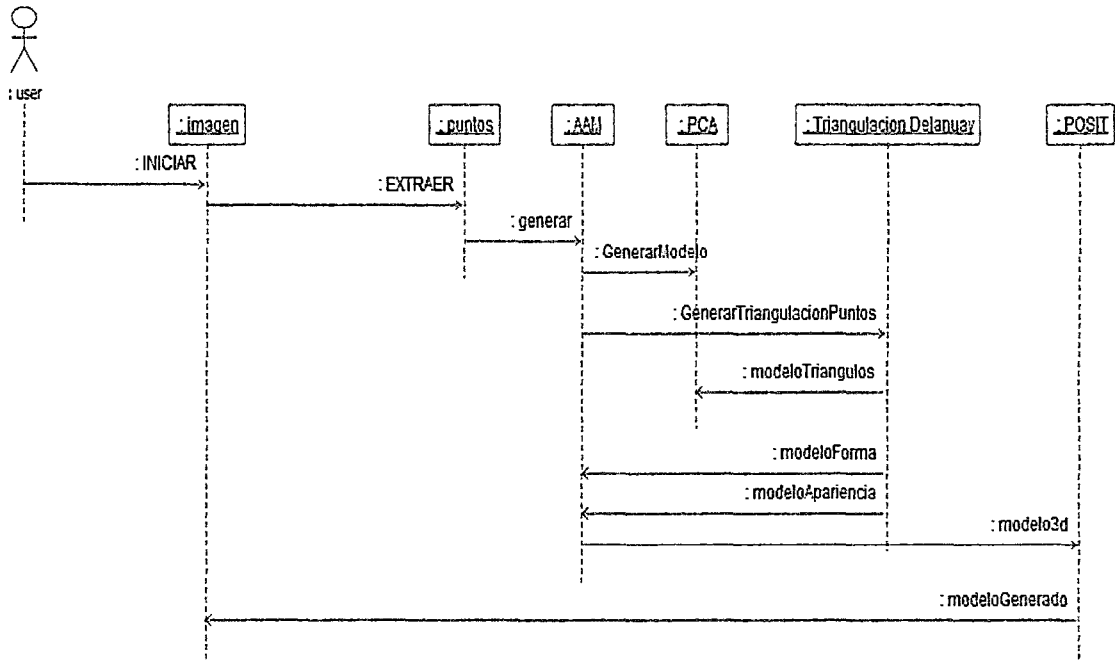
El software realiza diversas tareas antes de poder generar un modelo de Apariencia Activa y posteriormente utilizar este modelo para generar una vista en 3D a continuación se muestran los diagramas de secuencia y colaboración que nos dan un panorama de los módulos del software.

4.3.2.1. Diagrama de Secuencias

La FIGURA 16 muestra el diagrama de secuencias del software. En él se aprecia el intercambio de mensajes es decir la forma en que son invocados.

FIGURA 16

DIAGRAMA DE SECUENCIA SOFTWARE AAM-POSIT

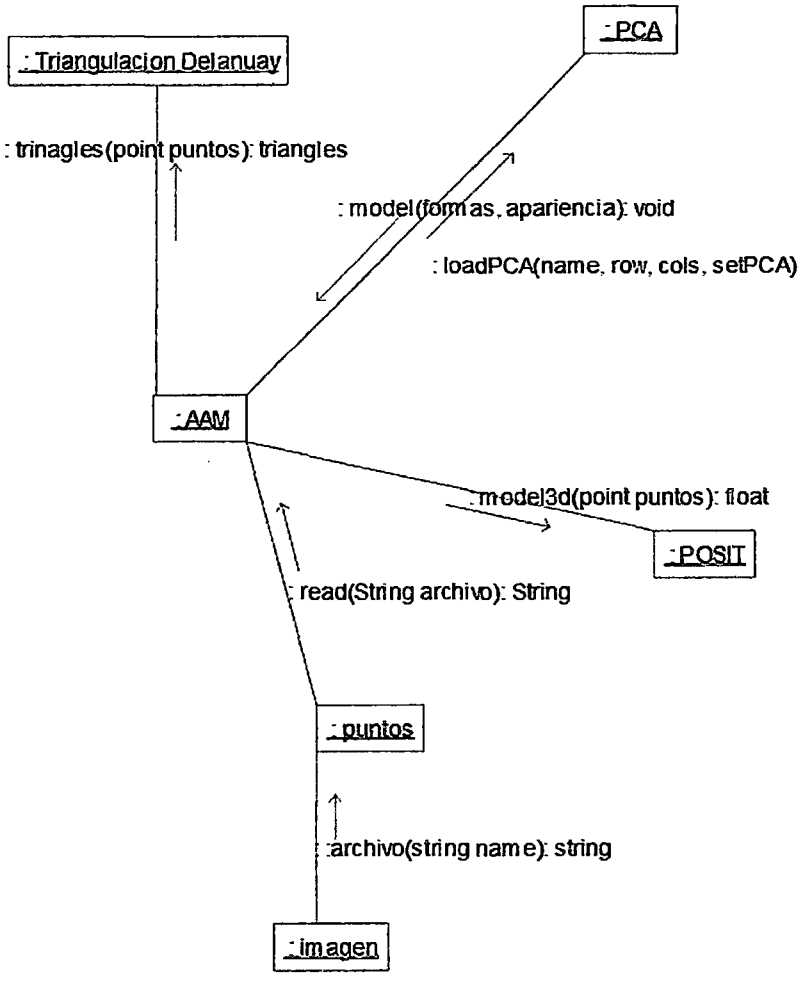


4.3.2.2. Diagrama de Colaboración

La FIGURA 17 muestra el diagrama de colaboración que nos muestran las interacciones que ocurren entre los objetos, esto nos permite fijar el interés en las relaciones entre los objetos y su topología

FIGURA 17.

DIAGRAMA DE COLABORACIÓN



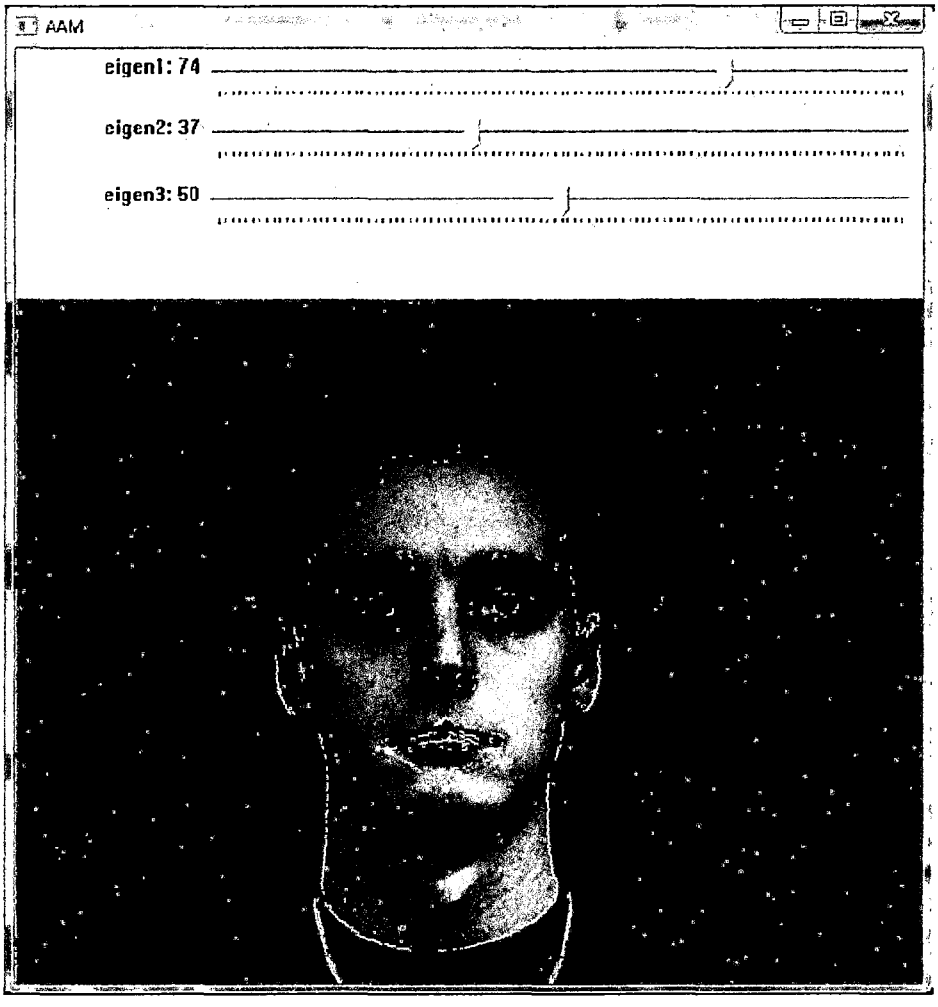
4.3.3. DISEÑO DE PROTOTIPO

Para el diseño de interfaces se desarrolló prototipos evolutivos ya que estos se basan en la idea de desarrollar una implementación inicial y poderla refinar iterativamente hasta lograr un producto maduro.

La FIGURA 18 muestra la captura de pantalla de la interfaz para la generación del modelo de apariencia activa y forma activa.

FIGURA 18

DISEÑO DE INTERFAZ PARA LA GENERACIÓN DE AAM Y ASM

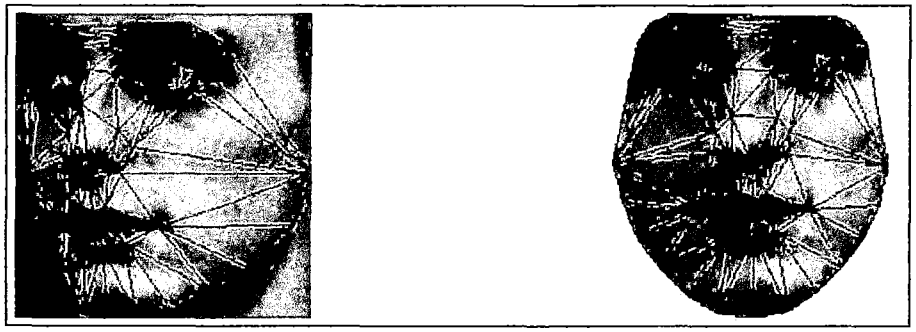


También en la FIGURA 18 se aprecia los puntos referenciales y la forma proyectada realizada con PCA utilizando todos los vectores propios

Luego la FIGURA 19 muestra la triangulación generada con el algoritmo Delanuay posterior a utilizar los puntos dados de la figura y los mapea en una nueva forma. Esta nueva forma es la proyección de las texturas sobre la malla generada con la triangulación Delanuay.

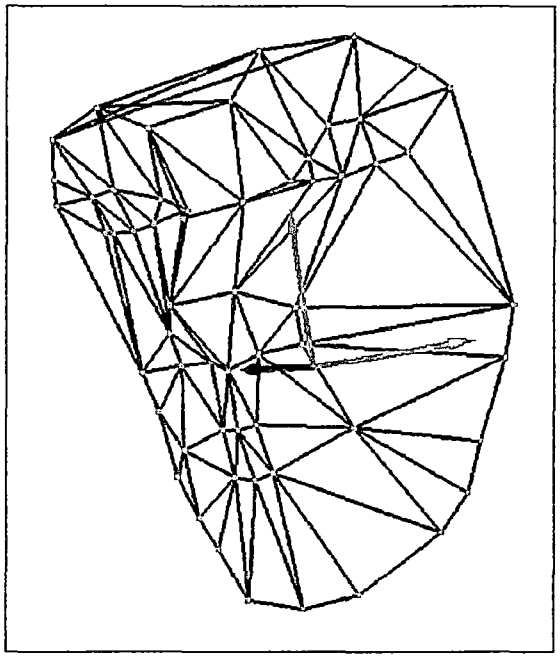
FIGURA 19

TRIANGULACIÓN Y PROYECCIÓN DE TEXTURA Y FORMA



Una vez generado el modelo de apariencia activa la siguiente tarea a realizar será generación del modelo 3D utilizando el algoritmo POSIT la FIGURA 20 muestra un modelo general de la cabeza humana en 3D.

FIGURA 20
PROTOTIPO DEL MODELO DE LA CABEZA HUMANA GENERADA EN 3D



4.4. IMPLEMENTACIÓN DEL SOFTWARE

Durante la investigación se realizó utilizando el lenguaje de programación C++, compilador GCC bajo el Sistema Operativo Ubuntu (Distribución de LINUX), IDE QTCreator y la librería de Visión por Computador OpenCV.

4.4.1. LECTURA DE PUNTOS

Este módulo desarrollado permite la lectura de los puntos anotados para ser cargados en al modelo generado con Análisis de Componentes Principales. El CUADRO 3 muestra el código desarrollado para la ejecución de esta tarea.

CUADRO 3

CODIGO FUENTE DE LA FUNCIÓN QUE IMPLEMENTA LA LECTURA DE LOS PUNTOS

```
PCA loadPCA(const char* fileName, int& rows, int& cols,Mat&
pcaset){
          FILE* in = fopen(fileName,"r");
          int a;
          fscanf(in,"%d%d",&rows,&cols);

          pcaset = Mat::eye(rows,cols,CV_64F);
          int i,j;
          i=j=0;

          for(i=0;i<rows;i++){
              for(j=0;j<cols;j++){
                  fscanf(in,"%d",&a);
                  pcaset.at<double>(i,j) = a;
              }
          }
          cout << pcaset << endl;
          PCA pca(pcaset,
               Mat(), ,
              CV_PCA_DATA_AS_ROW,
              pcaset.cols  );
          return pca;
}
```

FUENTE: CODIGO FUENTE DEL SOFTWARE

4.4.2. MODELO DE APARIENCIA ACTIVA

La función del modulo encargado de la generación del modelo de Apariencia Activa se denomina createAAM, el código fuente del modelo se muestra a continuación en el cuadro 4.

CUADRO 4

CODIGO FUENTE DE LA FUNCION QUE IMPLEMENTA LA GENERACIÓN DEL MODELO DE APARIENCIA ACTIVA

```
void createAAM(PCA pca, Mat pcaSet, PCA& pcaTexture,
std::vector<CvPoint>& pointsInsideHull,vector<int>& triangleVertices){
    CvMemStorage* storage;
    CvSubdiv2D* subdiv;
    CvRect rect = { 0, 0, 640, 480 };
    IplImage* asmFrame = cvCreateImage( cvSize (640,480),
IPL_DEPTH_32F,3);
storage = cvCreateMemStorage(0);
    subdiv = cvCreateSubdivDelaunay2D(rect,storage);
    std::vector<CvPoint> points;

    for(int i=0;i<pca.mean.cols/2;i++){
        double x = pca.mean.at<double>(0,2*i);
        double y = pca.mean.at<double>(0,2*i+1);
        CvPoint point = cvPoint( cvRound(x), cvRound(y));
        points.push_back(point);
        CvPoint2D32f fp = cvPoint2D32f(x, y);
        cvSubdivDelaunay2DInsert( subdiv, fp );
    }
    //crear convex hull
    CvPoint* pointsHull = (CvPoint*)malloc( points.size() *
sizeof(pointsHull[0]));
    int* hull = (int*)malloc( points.size() * sizeof(hull[0]));
    CvMat pointMat = cvMat( 1, points.size(), CV_32SC2, pointsHull );
    CvMat hullMat = cvMat( 1, points.size(), CV_32SC1, hull );

    for(int i = 0; i < points.size(); i++ )
    {
        pointsHull[i] = points.at(i);
    }
    cvConvexHull2( &pointMat, &hullMat, CV_CLOCKWISE, 0 );
    int hullcount = hullMat.cols;

    CvPoint* pointsHullFinal = (CvPoint*)malloc( hullcount *
sizeof(pointsHullFinal[0]));

    CvPoint pt0 = points[hull[hullcount-1]];

    for(int i = 0; i < hullcount; i++ ){
        CvPoint pt = points[hull[i]];
        pt0 = pt;
        pointsHullFinal[i] = pt0;
    }

    CvMat hullMatPoints = cvMat( 1, hullcount, CV_32SC2,
pointsHullFinal);
    for(int i=0;i< 640;i++){
        for (int j=0;j<480;j++){
            double distance =
cvPointPolygonTest(&hullMatPoints,cvPoint2D32f(i,j),1);
            if(distance >=0){
```

```
                pointsInsideHull.push_back(cvPoint(i,j));
            }
        }
    }
    int textureRows = pcaSet.rows;
    int textureCols = pointsInsideHull.size();
    Mat pcaTextureSet = Mat::eye(textureRows,textureCols*3,CV_64F);

    for(int imageIndex=0;imageIndex<3;imageIndex++){

        char imageFileName[200];
        sprintf(imageFileName,"09-%dm.jpg",imageIndex+1);
        IplImage* img = cvLoadImage(imageFileName);
        Mat matImgFrame(img);

        Mat warp_final;
        warp_final = Mat::zeros( matImgFrame.rows, matImgFrame.cols,
matImgFrame.type() );

draw_subdiv(asmFrame,subdiv,10,CV_NEXT_AROUND_LEFT,points,pcaSet,matImg
Frame,imageIndex,warp_final,triangleVertices);

draw_subdiv(asmFrame,subdiv,10,CV_NEXT_AROUND_RIGHT,points,pcaSet,matIm
gFrame,imageIndex,warp_final,triangleVertices);

        int pointIndex = 0;

        for(int j=0;j<textureCols;j++){
            CvPoint pt = pointsInsideHull.at(pointIndex);
            int pos = pt.y* img->widthStep + pt.x *3;

            pcaTextureSet.at<double>(imageIndex,3*j          )      =
((double)*((uchar*)(warp_final.data  + pos)))/255.0f;
            pcaTextureSet.at<double>(imageIndex,3*j+1)              =
((double)*((uchar*)(warp_final.data  + pos+1)))/255.0f;
            pcaTextureSet.at<double>(imageIndex,3*j+2)              =
((double)*((uchar*)(warp_final.data  + pos+2)))/255.0f;
            pointIndex++;
        }

        cvReleaseImage(&img);
        warp_final.release();
        matImgFrame.release();
    }

    pcaTexture= PCA(pcaTextureSet,
        Mat(),
        CV_PCA_DATA_AS_ROW,
        pcaTextureSet.cols
        );
}
```

FUENTE: CODIGO FUENTE DEL SOFTWARE

4.4.3. MODELO 3D BASADO EN POSIT

El modulo encargado de la generación del modelo en 3D utiliza el algoritmo POSIT, El cual tiene como entrada el modelo de la malla 2D generada en el módulo de apariencia activa que se realizó en el

módulo AAM. El cuadro 5 muestra parte del código fuente desarrollado para la generación del modelo 3D.

CUADRO 5

CODIGO FUENTE DE LA FUNCION QUE IMPLEMENTA LA GENERACIÓN DEL MODELO 3D BASADO en POSIT

```
Void Anthropometric3DModel(int LineWidth){
    int k;
    glColor3f(0.0, 0.0, 0.0);
    //glLineWidth(2.0);
    glLineWidth(LineWidth);
    glPushMatrix();
    glBegin(GL_LINES);
        for(k=0;k<95;k++){
glVertex3f(Model3D[MeanDelaunayTriangles[k][0]][0],Model3D[MeanD
elaunayTriangles[k][0]][1],Model3D[MeanDelaunayTriangles[k][0]][
2]);

glVertex3f(Model3D[MeanDelaunayTriangles[k][1]][0],Model3D[MeanD
elaunayTriangles[k][1]][1],Model3D[MeanDelaunayTriangles[k][1]][
2]);

glVertex3f(Model3D[MeanDelaunayTriangles[k][1]][0],Model3D[MeanD
elaunayTriangles[k][1]][1],Model3D[MeanDelaunayTriangles[k][1]][
2]);

glVertex3f(Model3D[MeanDelaunayTriangles[k][2]][0],Model3D[MeanD
elaunayTriangles[k][2]][1],Model3D[MeanDelaunayTriangles[k][2]][
2]);

glVertex3f(Model3D[MeanDelaunayTriangles[k][2]][0],Model3D[MeanD
elaunayTriangles[k][2]][1],Model3D[MeanDelaunayTriangles[k][2]][
2]);

glVertex3f(Model3D[MeanDelaunayTriangles[k][0]][0],Model3D[MeanD
elaunayTriangles[k][0]][1],Model3D[MeanDelaunayTriangles[k][0]][
2]);

        }
    glEnd();

        glColor3f(1.0, 0.0, 0.0);
        glPointSize(LineWidth+2);

        glBegin(GL_POINTS);
            for(k=0;k<58;k++)
glVertex3f(Model3D[k][0],Model3D[k][1],Model3D[k][2]);
        glEnd();

    glPopMatrix();
}
```

FUENTE: CODIGO FUENTE DEL SOFTWARE

4.5. EVALUACIÓN DEL SISTEMA

La evaluación del software se realizó a través de las características y sub-características especificadas según la norma ISO/IEC 9126-2 métricas de calidad interna de software y las especificaciones de la norma ISO/IEC 9126-3 métricas de calidad externa de software.

Los valores se hallaron por la sumatoria de las sub características y características como se especifica en la norma ISO/IEC 9126 como se detalla a continuación.

$$Vsc = \frac{\sum m}{n}$$

Donde:

Vsc: valor de la subcaracterística
m: valor de la métrica
n: número de métrica.

$$Vc = \frac{\sum mVsc}{nsc}$$

Donde:

Vc: valor de la característica
Vsc: valor de la subcaracterística
nsc: número de subcaracterística

La información de la calidad interna obtenida se resume a continuación en el cuadro 6.

CUADRO 6

CUADRO RESUMEN DE LA EVALUACIÓN DE LA CALIDAD INTERNA DEL SOTWARE

Característica	Sub característica	Métrica	m	Vsc	Vc	Valor total medio
Funcionalidad	Adecuación	Adecuación funcional	0.80	0.80	0.82	0.81
		Completitud de la implementación funcional	0.79			
	Seguridad	Acceso controlable	0.84	0.84		
Fiabilidad	Madurez	Detección de Fallas	0.50	0.75	0.75	
		Eliminación de fallas	1.00			
Usabilidad	Inteligibilidad	Funciones evidentes	0.90	0.87	0.87	
		Funciones entendibles	0.84			
	Atractividad	interacción atractiva	0.87	0.87		

Fuente: Elaboración Propia.

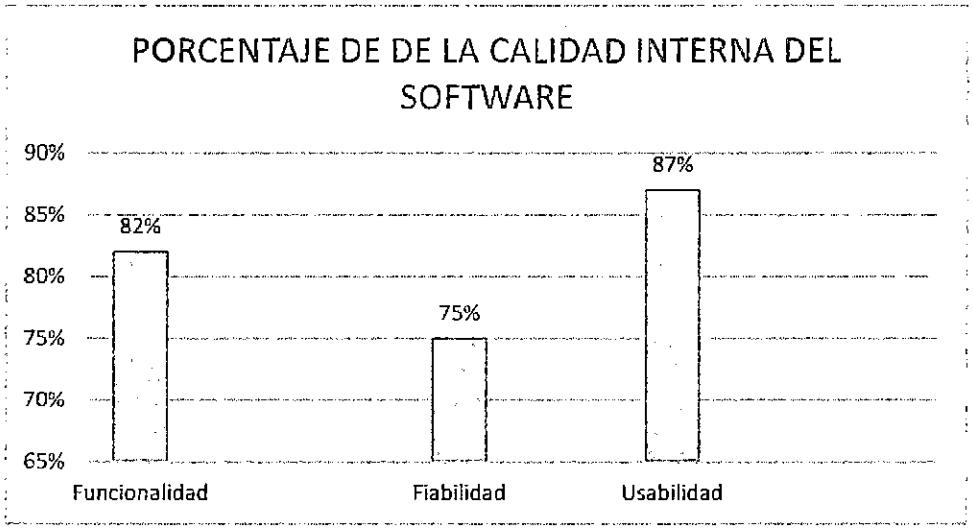
Nota: Elaborado en base a la norma ISO/IEC 9126-3

El porcentaje general obtenido para la métrica de calidad interna del software es del 81%, que indica que el software tiene atributos de funcionalidad, fiabilidad y usabilidad. Además se aprecia también que la evaluación individual del atributo funcionabilidad es del 82%, Fiabilidad 75% y de usabilidad 87%.

La figura 21 resume la información de la evaluación de las tres características en un gráfico de columnas

FIGURA 21

DIAGRAMA DE COLUMNAS PARA LA EVALUACIÓN DE CALIDAD DE SOFTWARE SEGÚN ISO/IEC 9126-3



El cuadro 7 muestra los resultados de la evaluación de las métricas de calidad externa según la ISO/IEC 9126-2 aplicados al software desarrollado.

CUADRO 7

CUADRO RESUMEN DE LA EVALUACION DE LA CALIDAD EXTERNA DEL SOFTWARE

Característica	Sub Característica	Métrica	m	Vsc	Vc	Valor Total Medio
Funcionalidad	Exactitud	Precisión de las expectativas	0.70	0.84	0.84	0.85
	Interoperabilidad	Intercambio de datos	0.98			
Fiabilidad	Madurez	Densidad de fracaso frente a casos de prueba	0.88	0.94	0.94	
		Resolución de fallas	1.00			
Eficiencia	Comportamiento en el tiempo	Tiempo de respuesta	0.80	0.78	0.78	
		tiempo de espera	0.75			
Mantenibilidad	Analizabilidad	apoyo a la función de diagnóstico	0.91	0.91	0.91	

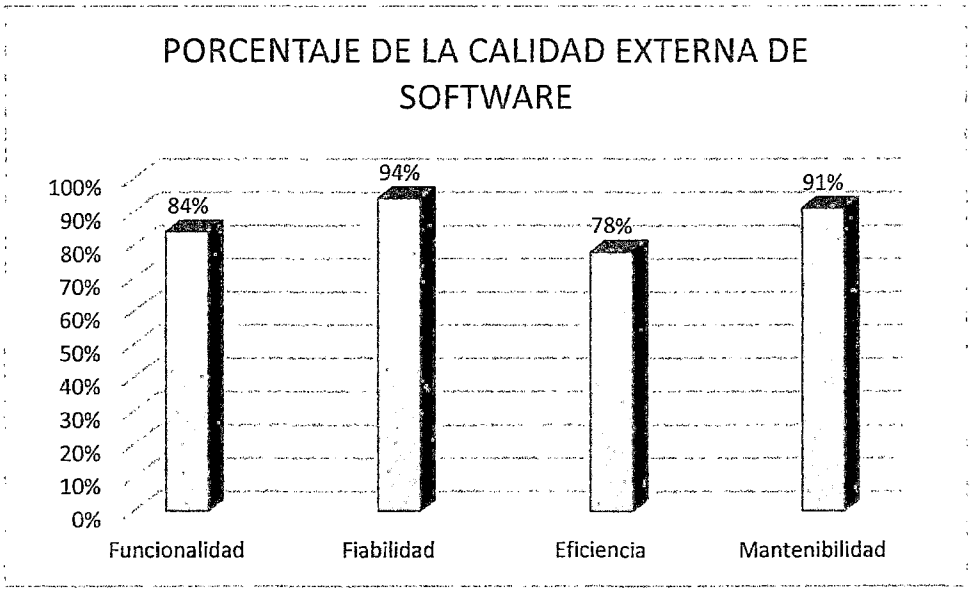
Fuente: Elaboración Propia.

Nota: Elaborado en base a la norma ISO/IEC 9126-4

Del cuadro anterior se observa que el porcentaje para la calidad externa es de un 88%, que indica que el software tiene atributos de exactitud, interoperabilidad, madurez, comportamiento en el tiempo y analizabilidad. En la figura 22 se aprecia el gráfico de columnas que resume los resultados de evaluación de las métricas del software.

FIGURA 22

DIAGRAMA DE COLUMNAS PARA LA EVALUACIÓN DE CALIDAD DE SOFTWARE SEGÚN ISO/IEC 9126-3



4.6. PRUEBA DE HIPOTESIS

Para la contrastación de hipótesis se utilizó el estadístico de prueba z debido a que la cantidad de ensayos realizados con el software es superior a 30. Se formularon las siguientes hipótesis:

H_0: El software desarrollado no tiene una eficiencia aceptable en la estimación automática de la posición de la cabeza mediante el uso de Active Apareance Model y POSIT.

H_a: El software desarrollado tiene una eficiencia aceptable en la estimación automática de la posición de la cabeza mediante el uso de Active Apareance Model y POSIT.

El software tiene una eficiencia aceptable en la estimación automática de la posición de la cabeza mediante el uso de Active Apareance Model y POSIT.

La hipótesis Estadística a probar fueron:

$$H_0: \pi = 75\%$$

$$H_a: \pi \neq 75\%$$

Datos

$$\alpha = 0.05$$

$$n = 33$$

$$p = 79\%$$

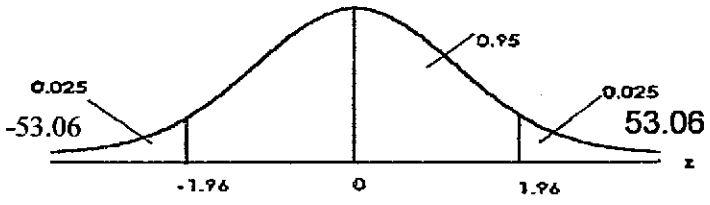
$$\pi = 75\%$$

Calculo del valor de z

$$Z = \frac{p - \pi_H}{\sigma_p}$$

$$z = \frac{79 - 75}{\sqrt{\frac{0.75 * 0.25}{33}}} = 53.06$$

Gráfica de la región de aceptación



Contrastación con Z de tabla

Como el valor de Z calculada es igual a 53.06 mayor a Z de Tabla al 95% de probabilidad debemos de rechazar la hipótesis nula y aceptar la alterna, la proporción obtenida es mayor al 75%.

Conclusión

Podemos concluir que según la prueba de hipótesis para la proporción que el software desarrollado tiene una eficiencia aceptable en la estimación automática de la posición de la cabeza mediante el uso de Active Apareance Model y POSIT.

CONCLUSIONES

Se realizó la identificación de los requerimientos de usuario recolectándose un total de 11 historias de usuario que fueron útiles para el desarrollo del software para la estimación de la cabeza humana

El análisis y diseño del software permitió especificar un total de 15 tareas de ingeniería que permitieron el desarrollo de los módulos. Además el análisis de los casos de uso, los diagramas de secuencia y los diagramas de colaboración fueron útiles para la implementación del software para la estimación de la posición de la cabeza.

La implementación del software se desarrolló basado en las especificaciones y el análisis, esto permitió el desarrollo de los módulos para la triangulación de Delanuay, análisis de componentes principales, modelo de apariencia activa, lectura de data, lectura de imagen y la proyección en 3D del modelo generado basado en POSIT.

La metodología para evaluar la calidad del software para la estimación de la cabeza humana basado en Active Apareance Model y el algoritmo POSIT fue el ISO/IEC 9126 esta permitió determinar un conjunto de características que

cumplen con los requisitos especificados por el usuario logrando una calidad interna del 81 % y una calidad externa del 85%. Además la evaluación de la hipótesis General mediante la prueba Z para la contratación de hipótesis de la proporción dio como resultado que el software tiene una eficiencia aceptable en la estimación de la posición de la cabeza humana mediante el uso de AAM y POSIT.

RECOMENDACIONES

Se recomienda el uso de los modelos estadísticos para la descripción de las formas y texturas específicamente los modelos de forma activa y los modelos de apariencia activa aplicadas a imágenes en 2 dimensiones, debido a que permiten el almacenamiento de las características de los rostros de una manera eficiente.

Se recomienda el uso del algoritmo POSIT para la proyección de un modelo 2D a un modelo 3D por sus cualidades de rápida convergencia a un modelo base.

Se recomienda el uso de las tareas de usuario propias de la metodología XP para la identificación de los requerimientos de usuario, el uso de las tareas de ingeniería para la especificación de los requisitos del software.

Finalmente se recomienda el uso de la Norma ISO/IEC 9126 para la evaluación de los productos de software.

BIBLIOGRAFÍA

Cootes, T. F., Edwards, G. J., & Taylor, C. J. (1998). Active Appearance Models. *ECCV 2*, (págs. 484-498).

Cootes, T. F., Taylor, C. J., Cooper, D. H., & Graham, J. (1995). Active Shape Models . *Computing Vision and Image Understanding*, 38-59.

Dias Martins, P. A. (2008). *Active Appearance Models for Facial Expression Recognition and Monocular Head Pose Estimation.* Coimbra: University of Coimbra.

Gerbrands, J. (1981). On the relationships between SVD, KLT and PCA. *Pattern Recognition*, 375-381.

Gonzales. (2007). *Digital Image Processing.* Pearson Education International.

Haider, A., & Kaneko, T. (2000). *Realistic 3D Head Modeling from Video Captured Imafes and CT Data.* Toyohashi University of Technology.

Loeve, M. (1955). *Probability Theory.* Princeton: Van Nostrand.

Murugan M., V., & Mathews M., S. (2013). 2D y 3D Active Shape Model with SURF algoritm for Object Retrieval. *International Conference on Advanced Computing and Communication Systems*, (págs. 19-21). Coimbatore.

Valenti, R., Sebe, N., & Gevers, T. (2012). Combining Head Pose and Eye Location Information for Gaze Estimation. *IEEE TRANSACTIONS ON IMAGE PROCESSING VOL 21, NO. 2*, 802-815.

Van Ginneken, B., Frangi, A., Stall, J., & Ter Haar Romeny, B. (2002). Active shape model segmentation with optimal features. *IEEE Trans. Med. Imaging*, 924-933.

Whitehill, J., & Movellan, J. (2008). A discriminative approach to frame-by-frame head pose tracking. *Automatic Face & Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on* (págs. 1-7). Amsterdam: IEEE.

ANEXOS

ANEXO 1 ISO/IEC TR 9126-3 MÉTRICAS INTERNAS

Table 8.1.1 Suitability metrics

Internal suitability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Sources of input measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional adequacy	How adequate are the checked functions?	Count the number of implemented functions that are suitable for performing the specified tasks, then measure the ratio of it to functions implemented. The following may be measured; -all or parts of design specifications -completed modules/parts of software products	X=1-A/B A= Number of functions in which problems are detected in evaluation B= Number of functions checked	0 <= X <= 1 The closer to 1, the more adequate.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers
Functional implementation completeness	How complete is the functional implementation?	Count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications	X=1-A/B A=Number of missing functions detected in evaluation. B=Number of functions described in requirement specifications ***NOTE**: Input to the measurement process is the updated requirement specifications. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.*	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers

Internal suitability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Sources of input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional implementation coverage	How correct is the functional implementation?	Count the number of incorrectly implemented or missing functions and compare with the number of functions described in the requirement specifications ***Note:*** *Review by functional item.*	X=1-A/B A= Number of incorrectly implemented or missing functions detected. B= Number of functions described in requirement specifications ***Note:*** *Input to the measurement process is the updated requirement specifications. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.*	0 <= X <= 1 The closer to 1, the more correct.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers
Functional specification stability (volatility)	How stable is the functional specification during the development life cycle?	Count the number of functions changed (added, modified, or deleted) during development life cycle phase, then compare with the number of functions described in the requirement specifications.	X=1-A/B A=Number of functions changed during development life cycle phases B=Number of functions described in requirement specifications	0 <= X <= 1 The closer to 1 the more stable.	absolute	A=Count B=Count X=Count/ Count	Requirement specifications Review report	6.5 Validation 6.3 Quality Assurance 5.3 Qualification testing 6.8 Problem Resolution 5.4 Operation	Developers Maintainers

Table 8.1.2 Accuracy metrics

Internal accuracy metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Computational Accuracy	How completely have the accuracy requirements been implemented?	Count the number of functions that have implemented the accuracy requirements and compare with the number of functions with specific accuracy requirements.	X=A/B A= Number of functions in which specific accuracy requirements had been implemented, as confirmed in evaluation. B= Number of functions for which specific accuracy requirements need to be implemented.	0 <= X <= 1. The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	Verification Joint review	Requirers Developers
Precision	How complete was the implementation of specific levels of precision for the data items?	Count the number of data items that meet the requirements of specific levels of precision and compare to the total number of data items with specific level of precision requirements.	X=A/B A= Number of data items implemented with specific levels of precision, confirmed in evaluation B= Number of data items that require specific levels of precision	0 <= X <= 1. The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	Verification Joint review	Requirers Developers

Table 8.1.3 Interoperability metrics

Internal interoperability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Data exchangeability (Data format based)	How correctly have the interface data formats been implemented?	Count the number of interface data formats that have been implemented correctly as in the specifications and compare to the number of data formats to be exchanged as in the specifications.	X=A/B A=Number of interface data formats that have been implemented correctly as in the specifications B=Number of data formats to be exchanged as in the specifications	0 <= X <= 1. The closer to 1, the more correct.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	Verification Joint review	Developers Requirers
Interface consistency (protocol)	How correctly have the interface protocols been implemented?	Count the number of interface protocols that were implemented correctly as in the specifications and compare with the number of interface protocols to be implemented as in the specifications.	X=A/B A=Number of interface protocols implementing consistent format as in the specification confirmed in review B=Number of interface protocols to be implemented as in the specifications	0 <= X <= 1 The closer to 1, the more consistent.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	Verification Joint review	Developers Requirers

Table 8.1.4 Security metrics

Internal security metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Access auditability	How auditable is access login?	Count the number of access types that are being logged correctly as in the specifications and compare with the number of access types that are required to be logged in the specifications.	X=A/B A= Number of access types that are being logged as in the specifications B= Number of access types required to be logged in the specifications	0 <= X <= 1 The closer to 1, the more auditable.	Absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers
Access controllability	How controllable is access to the system?	Count the number of access controllability requirements implemented correctly as in the specifications and compare with the number of access controllability requirements in the specifications.	X=A/B A= Number of access controllability requirements implemented correctly as in the specifications. B= Number of access controllability requirements in the specifications..	0 <= X <= 1 The closer to 1, the more controllable.	Absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers

Internal security metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Data corruption prevention	How complete is the implementation of data corruption prevention?	Count the number of implemented instances of data corruption prevention as specified and compare with the number of instances of operations/ access specified in requirements as capable of corrupting/ destroying data.	X=A/B A= Number of implemented instances of data corruption prevention as specified confirmed in review. B= Number of instances of operation/access identified in requirements as capable of corrupting/destroying data Note: Consider security levels when using this metric.	0 <= X <= 1 The closer to 1, the more complete.	Absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation 6.6 Joint review	Developers
Data encryption	How complete is the implementation of data encryption?	Count the number of implemented instances of encryptable/decryptable data items as specified and compare with the number of instances of data items requiring data encryption/decryption facility as in specifications.	X=A/B A=Number of implemented instances of encryptable/decryptable data items as specified confirmed in review B= Number of data items requiring data encryption/decryption facility as in specifications ***NOTE:*** *Data encryption: e.g., data in open database, data in public communication facility*	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation	Developers

Table 8.1.5 Functionality compliance metrics

Internal functionality compliance metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional compliance	How compliant is the functionality of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.	X=A/B A= Number of correctly implemented items related to functionality compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1. The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers
Intersystem standard compliance	How compliant are the interfaces to applicable regulations, standards and conventions	Count the number of interfaces that meet required compliance and compare with the number of interfaces requiring compliance as in the specifications ***Note:** All specified attributes of a standard must be checked*	X=A/B A= Number of correctly implemented interfaces as specified, confirmed in review B= Total number of interfaces requiring compliance	0 <= X <= 1. The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	Verification Joint review	Developers Requirers

Table 8.2.1 Maturity metrics

Internal maturity metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Fault detection ***Note:*** *this metric should only be used for prediction during development.*	How many faults were detected in reviewed product?	Count the number of detected faults in review and compare it to the number of estimated faults to be detected in this phase.	X=A/B A=Absolute number of faults detected in review B=Number of estimated faults to be detected in review (using past history or reference model)	0 <= X A high value for X implies good product quality, while A=0 does not necessarily imply fault free status of the reviewed item. ***NOTE:*** *1. It is necessary to convert this value(X) to the <0,1> interval if making summarization of characteristics.*	Absolute	X=count/count A=count B=count	Value A comes from review report Value B comes from the organizatio n database.	Verification Joint review	Requirers Developers

Internal maturity metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Fault removal	How many faults have been corrected? What is the proportion of faults removed?	Count the number of faults removed during design/coding and compare it to the number of faults detected in review during design/coding.	X=A A=Number of corrected faults in design/coding Y=A/B A=Number of corrected faults design/coding B= Number of faults detected in review	0 <= X A high value of X implies, that less faults remain. 0 <= Y <= 1 The closer to 1, the better. (more faults removed) ***NOTE:*** *1. It is necessary to convert this value (X) to the <0,1> interval if making summarization of characteristics.*	ratio absolute	X=count A=count Y=count/count B=count	Value A comes from fault removal report. Value B comes from review report.	Verification Joint review	Requirers Developers
Test adequacy	How much of the required test cases are covered by the test plan?	Count the number of test cases planned and compare it to the number of test cases required to obtain adequate test coverage.	X=A/B A=Number of test cases designed in test plan and confirmed in review B= Number of test cases required	0 <= X Where X is greater the better adequacy	absolute	X=count/count A=count B=count	Value A comes from test plan Value B comes from requirements	QA Problem resolution Verification	Developers Maintainers

Table 8.2.2 Fault tolerance metrics

Internal fault tolerance metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Failure avoidance	How many fault patterns were brought under control to avoid critical and serious failures?	Count the number of avoided fault patterns and compare it to the number of fault patterns to be considered	X=A/B A=Number of fault patterns having avoidance in design/code B=Number of fault patterns to be considered ***NOTE:*** *Fault pattern examples* *out of range data* *deadlock* ***NOTE:*** *Fault tree analysis technique may be used to detect fault patterns.*	0 <= X Where X is greater the better failure avoidance	absolute	X=count/count A=count B=count	Value A comes from review report Value B comes from requirement specification document.	Verification Validation Joint review Problem resolution	Developers Requirers Maintainers
Incorrect operation avoidance	How many functions are implemented with incorrect operations avoidance capability?	Count the number of implemented functions to avoid critical and serious failures caused by incorrect operations and compare it to the number of incorrect operation patterns to be considered. ***NOTE:*** *Also data damage in addition to system failure.*	X=A/B A=Number of functions implemented to avoid incorrect operation patterns. B=Number of incorrect operation patterns to be considered ***NOTE:*** *Incorrect operation patterns* *Incorrect data types as parameters* *Incorrect sequence of data input* *Incorrect sequence of operation* ***NOTE:*** *Fault tree analysis technique may be used to detect incorrect operation patterns.*	0 <= X Where X is greater the better incorrect operation avoidance.	absolute	X=count/count A=count B=count	Value A comes from review report Value B comes from requirement specification document.	Verification Validation Joint review Problem resolution	Developers Requirers Maintainers

Table 8.2.3 Recoverability metrics

Internal recoverability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Target audience
Restorability	How capable is the product in restoring itself after abnormal event or at request?	Count the number of implemented restoration requirements and compare it to the number of restoration requirements in the specifications. Restoration requirement examples: database checkpoint, transaction checkpoint, redo function, undo function	X=A/B A=Number of implemented restoration requirements confirmed in review B=Number of restoration requirements in the specifications..	0 <= X <= 1 Where X is greater, the better restorability	Absolute	X=count/count A=count B=count	A comes from review document B comes from requirements or design document	Verification Joint review	Developers Maintainers
Restoration Effectiveness	How effective is the restoration capability?	Count the number of implemented restoration requirements meting target restoration time (by calculations or simulations) and compare it to the number of restoration requirements with specified target time.	X=A/B A=Number of implemented restoration requirements meeting target restore time B=Number of restoration requirements with specified target times	0 <= X <= 1 Where X is greater, the better effectiveness	Absolute	X=count/count A=count B=count	A comes from review document B comes from requirements or design document.	Verification Joint review	Developers Maintainers

Table 8.2.4 Reliability compliance metrics

Internal reliability compliance metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Reliability compliance	How compliant is the reliability of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification..	X=A/B A= Number of correctly implemented items related to reliability compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1. The closer to 1, the more compliant.	Absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers

Table 8.3.1 Understandability metrics

Internal understandability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Completeness of description	What proportion of functions (or types of function) are described in the product description?	Count the number of functions which are adequately described and compare with the total number of functions in the product.	X= A/B A= Number of functions (or types of functions) described in the product description B= Total number of functions (or types of functions)	0<=X<=1 The closer to 1 the more complete	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
***NOTE 1**: This indicates whether potential users will understand the capability of the product after reading the product description.* ***NOTE 2**: See also ISO/IEC 9127 Consumer software package.*									
Demonstration capablity	What proportion of functions requiring demonstration have demonstration capability?	Count the number of functions that are adequately demonstrable and compare with the total number of functions requiring demonstration capability	X=A/B A= Number of functions demonstrated and confirmed in review B= Total number of functions requiring demonstration capability	0<=X<=1 The closer to 1 the more capable.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
***NOTE**: Demonstrations step through the process showing how the product is used. This includes "wizards".*									
Evident functions	What proportion of the product functions are evident to the user?	Count the number of functions that are evident to the user and compare with the total number of functions	X= A/B A= Number of functions (or types of functions) evident to the user B= Total number of functions (or types of functions)	0<=X<=1 The closer to 1 the better	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
***NOTE**: This indicates whether users will be able to locate functions by exploring the interface (e.g. by inspecting the menus)*									
Function understandability	What proportion of the product functions will the user be able to understand correctly.	Count the number of user interface functions where purposes is understood by the user and compare with the number of user interface functions.	X= A/B A= Number of user interface functions whose purpose is understood by the user B= Number of user interface functions.	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers

Table 8.3.2 Learnability metrics

Internal learnability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Completeness of user documentation and/or help facility	What proportion of functions are described in the user documentation and/or help facility?	Count the number of functions implemented with help facility and/or documentation and compare with the total number of functions in product.	X= A/B A= Number of functions described B= Total of number of functions provided	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers

***NOTE:** : Three metrics are possible: completeness of the documentation, completeness of the help facility or completeness of the help and documentation used in combination.*

Table 8.3.3 Operability metrics Internal Operability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Input validity checking	What proportion of input items provide check for valid data	Count the number of input items, which check for valid data and compare with the number of input items, which could check for valid data	X=A/B A=Number of input items which check for valid data B=Number of input items which could check for valid data	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
User operation cancellability	What proportion of functions can be cancelled prior to completion?	Count the number of implemented functions, which can be cancelled by the user prior to completion and compare it with the number of functions requiring the precancellation capability	X=A/B A=Number of implemented functions which can be cancelled by the user B= Number of functions requiring the precancellation capability	0 <= X <= 1 The closer to 1, the better cancellability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
User operation Undoability	What proportion of functions can be undone?	Count the number of implemented functions, which can be undone by the user after completion and compare it with the number of functions	X=A/B A=Number of implemented functions which can be undone by the user B= Number of functions.	0 <= X <= 1 The closer to 1, the better undoability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
***NOTE:** : Either single undoability or multiple undoability after several subsequent actions can be assessed*									
Customisability	What proportion of functions can be customised during operation?	Count the number of implemented functions, which can be customized by the user during operation and compare it with the number of functions requiring the customization capability	X=A/B A=Number of functions which can be customised during operation B=Number of functions requiring the customization capability	0 <= X <= 1 The closer to 1, the better customisability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
Physical accessibility	What proportion of functions can be customised for access by users with physical handicaps	Count the number of implemented functions, which can be customised and compare it with the number of functions	X=A/B A=Number of functions which can be customised B=Number of functions	0 <= X <= 1 The closer to 1, the better physical accessibility	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers

***NOTE:** Examples of physical accessibility are inability to use a mouse and blindness*

Operation status monitoring capability	What proportion of functions have operations status monitoring capability?	Count the number of implemented functions, which status can be monitored and compare it with the number of functions requiring the monitoring capability.	X=A/B A=Number of functions having status monitoring capability B=Number of functions that are required to have monitoring capability.	0 <= X <= 1 The closer to 1, the better monitoring capability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: : *Status includes progress monitoring.*									
Operational consistency	What proportion of operations behave the same way to similar operations in other parts of the system?	Count the number of instances of operations with inconsistent behaviour and compare it with the total number of operations	X=1 - A/B A=Number of instances of operations with inconsistent behaviour B=Total number of operations	0 <= X <= 1 The closer to 1, the more consistent	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
Message Clarity	What proportion of messages are self-explanatory?	Count the numbers of implemented messages with clear explanations and compare it with the total number of messages implemented.	X=A/B A=Number of implemented messages with clear explanations. . B=Number of messages implemented	0 <= X <= 1 The closer to 1, the more clear.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: : *Clear error messages explain to the user what action to take to recover from the error*									
Interface element clarity	What proportion of interface elements are self-explanatory?	Count the number of interface elements which are self explanatory and compare it with the total number of interface elements	X=A/B A=Number of interface elements which are self-explanatory. B=Total number of interface elements	0 <= X <= 1 The closer to 1, the more clear.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: : *Elements are self explanatory when they use plain text or provide "hover-help" or "tool tips"*									
Operational error recoverability	What proportion of functions can tolerate user error?	Count the number of functions implemented with user error tolerance and compare it to the total number of functions requiring the tolerance capability	X=A/B A=Number of functions implemented with user error tolerance B=Total number of functions requiring the tolerance capability	0 <= X <= 1 The closer to 1, the more recoverable.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers

Table 8.3.4 Attractiveness metrics

Internal attractiveness metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Attractive interaction	How attractive is the interface to the user?	Questionnaire to users	Questionnaire to assess the attractiveness of the interface to users, taking account of attributes such as colour and graphical design. NOTES: Issues that potentially contribute to attractiveness include: Alignment of items (vertical and Horizontal), Grouping, Use of colours, Appropriate and reasonable sized graphics, Use of whitespace/separators/borders, Animation, Typography, and 3D interface.	Assessment classification	Ordinal	X= Count (Count is a score)	Req spec Design Review report	Verification Joint review	Requirers Developers
NOTE: *This could be based on screen sketches or mock-ups*									
User Interface appearance customisability	What proportion of user interface elements can be customised in appearance.	Inspection (by expert)	X=A/B A=Number of types of interface elements that can be customised. B=Total number of types of interface elements.	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
NOTE:									

Table 8.3.5 Usability compliance metrics

Internal usability compliance metrics									
Metric name	Purpose	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Usability compliance	How compliant is the product to applicable regulations, standards and conventions for usability	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification..	X=A/B A= Number of correctly implemented items related to usability compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1 The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers
NOTE:									

Anexo 2 ISO/IEC TR 9126-3 Métricas Externas.

Table 8.1.1 Suitability metrics

External suitability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Sources of input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional adequacy	How adequate are the evaluated functions?	Number of functions that are suitable for performing the specified tasks comparing to the number of function evaluated.	X=1-A/B A= Number of functions in which problems are detected in evaluation B= Number of functions evaluated	0 <= X <= 1 The closer to 1.0, the more adequate.	Absolute	X= Count/ Count A= Count B= Count	Requirement specification (Req. Spec.) Evaluation report	6.5 Validation, 6.3 Quality Assurance, 5.3 Qualification testing	Developer, SQA
Functional implementation completeness	How complete is the implementation according to requirement specifications?	Do functional tests (black box test) of the system according to the requirement specifications. Count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications.	X = 1 - A / B A = Number of missing functions detected in evaluation B = Number of functions described in requirement specifications	0<=X<=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Req. spec. Evaluation report	6.5 Validation, 6.3 Quality Assurance, 5.3 Qualification testing	Developer, SQA

***NOTE**: 1. Input to the measurement process is the updated requirement specification. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.* *2. This metric is suggested as experimental use.*

NOTE: Any missing function can not be examined by testing because it is not implemented. For detecting missing functions, it is suggested that each function stated in a requirement specification be tested one by one during functional testing. Such results become input to "Functional implementation completeness" metric. For detecting functions which are implemented but inadequate, it is suggested that each function be tested for multiple specified tasks. Such results become input to the "Functional adequacy" metric. Therefore, users of metrics are suggested to use both these metrics during functional testing.

External suitability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Sources of input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional Implementation coverage	How correct is the functional implementation?	Do functional tests (black box test) of the system according to the requirement specifications. Count the number of incorrectly implemented or missing functions detected in evaluation and compare with the total number of functions described in the requirement specifications Count the number of functions that are complete versus the ones that are not.	X=1- A / B A= Number of incorrectly implemented or missing functions detected in evaluation B= Number of functions described in requirement specifications	0<=X<=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Req. spec. Evaluation report	6.5 Validation, 6.3 Quality Assurance, 5.3 Qualificatio n testing	Developer, SQA
***NOTE**: 1. Input to the measurement process is the updated requirement specification. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.* *2. This measure represents a binary gate checking of determining the presence of a feature.*									
Functional specification stability (volatility)	How stable is the functional specification after entering operation?	Count the number of functions described in functional specifications that had to be changed after the system is put into operation and compare with the total number of functions described in the requirement specifications.	X = 1- A / B A= Number of functions changed after entering operation starting from entering operation B= Number of functions described in requirement specifications	0<=X<= 1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Size	Req. spec. Evaluation report	6.8 Problem Resolution 5.4 Operation	Maintainer SQA
***NOTE**: This metric is suggested as experimental use.*									

Table 8.1.2 Accuracy metrics

External accuracy metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Accuracy to expectation	Are differences between the actual and reasonable expected results acceptable?	Do input .vs. output test cases and compare the output to reasonable expected results. Count the number of cases encountered by the users with an unacceptable difference from reasonable expected results.	X=A / T A= Number of cases encountered by the users with a difference against to reasonable expected results beyond allowable T= Operation time	0<=X The closer to 0 is the better.	Ratio	A= Count T= Time X= Count/ Time	Req. spec. User operation manual Hearing to users Test report	6.5 Validation 6.3 Quality Assurance	Developer User
***NOTE**: Reasonable expected results might be identified in a requirement specification, a user operation manual, or users' expectations.*									
Computational Accuracy	How often do the end users encounter inaccurate results?	Record the number of inaccurate computations based on specifications.	X=A / T A= Number of inaccurate computations encountered by users T= Operation time	0<=X The closer to 0 is the better.	Ratio	A= Count T= Time X= Count/ Time	Req. spec. Test report	6.5 Validation 6.3 Quality Assurance	Developer User
Precision	How often do the end users encounter results with inadequate precision ?	Record the number of results with inadequate precision.	X=A / T A= Number of results encountered by the users with level of precision different from required T= Operation time	0<=X The closer to 0 is the better.	Ratio	A= Count T= Time X= Count/ Time	Req. spec. Test report	6.5 Validation 6.3 Quality Assurance	Developer User

NOTE: Data elements for computation of external metrics are designed to use externally accessible information, because it is helpful for end users, operators, maintainers or acquirers to use external metrics. Therefore, the time basis metric often appears in external metrics and is different from internal ones.

Table 8.1.3 Interoperability metrics

External interoperability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Data exchangeability (Data format based)	How correctly have the exchange interface functions for specified data transfer been implemented?	Test each downstream interface function output record format of the system according to the data fields specifications. Count the number of data formats that are approved to be exchanged with other software or system during testing on data exchanges in comparing with the total number.	X= A / B A= Number of data formats which are approved to be exchanged successfully with other software or system during testing on data exchanges, B= Total number of data formats to be exchanged	0<=X<= 1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Req. spec. (User manual) Test report	6.5 Validation	Developer
NOTE: It is recommended to test specified data transaction.									
Data exchangeability (User's success attempt based)	How often does the end user fail to exchange data between target software and other software? How often are the data transfers between target software and other software successful? Can user usually succeed in exchanging data?	Count the number of cases that interface functions were used and failed.	a) X= 1 - A / B A= Number of cases in which user failed to exchange data with other software or systems B= Number of cases in which user attempted to exchange data b) Y= A / T T= Period of operation time	0<=X<= 1 The closer to 1.0 is the better. 0<=Y The closer to 0, is the better.	a) Absolute b) Ratio	A= Count B= Count X= Count/ Count Y= Count/ Time T= Time	Req. spec. (User manual) Test report	5.4 Operation	Maintainer

Table 8.1.4 Security metrics

External security metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Access auditability	How complete is the audit trail concerning the user access to the system and data?	Evaluate the amount of accesses that the system recorded in the access history database.	X= A / B A= Number of "user accesses to the system and data" recorded in the access history database B= Number of " user accesses to the system and data" done during evaluation	0<=X<=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Test spec. Test report	6.5 Validation	Developer

NOTE:*1. Accesses to data may be measured only with testing activities.*
2. This metric is suggested as an experimental use.
3. It is recommended that penetration tests be performed to simulate attacks, because such security attacks do not normally occur in the usual testing. Real security metrics may only be taken in "real life system environment", that is "quality in use".
4. "User access to the system and data" record may include "virus detection record" for virus protection. The aim of the concept of computer virus protection is to create suitable safeguards with which the occurrence of computer viruses in systems can be prevented or detected as early as possible.

Access controllability	How controllable is access to the system?	Count number of detected illegal operations with comparing to number of illegal operations as in the specification.	X= A / B A= Number of detected different types of illegal operations B= Number of types of illegal operations as in the specification	0<=X<=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Test spec. Test report Operation report	6.5 Validation 6.3 Quality Assurance	Developer

NOTE: *1. If it is necessary to complement detection of unexpected illegal operations additional intensive abnormal operation testing should be conducted.*
2. It is recommended that penetration tests be performed to simulate attack, because such security attacks do not normally occur in the usual testing. Real security metrics may only be taken in "real life system environment", thet is "quality in use".
3. Functions prevent unauthorized persons from creating, deleting or modifying programs or information. Therefore, it is suggested to include such illegal operation types in test cases.

External security metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Data corruption prevention	What is the frequency of data corruption events?	Count the occurrences of major and minor data corruption events.	a) X= 1 – A / N A= Number of times that a major data corruption event occurred N= Number of test cases tried to cause data corruption event b) Y= 1- B / N B= Number of times that a minor data corruption event occurred c) Z= A / T or B / T T= period of operation time (during operation testing)	0<=X<= 1 The closer to 1.0 is the better. 0<=Y<= 1 The closer to 1.0 is the better. 0<=Z The closer to 0, is the better.	a) Absolute b) Absolute c) Ratio	A= Count B= Count N= Count X= Count/ Count Y= Count/ Count T= Time Z= Count/ Time	Test spec. Test report Operation report	6.5 Validation 5.3 Qualifica-tion testing 5.4 Operation	Maintainer Developer

NOTE: *1. Intensive abnormal operation testing is needed to obtain minor and major data corruption events.*
2. It is recommended to grade the impact of data corruption events such as the following examples:
Major (fatal) data corruption event:
- reproduction and recovery impossible;
- second affection distribution too wide;
- importance of data itself.
Minor data corruption event:
- reproduction or recovery possible and
- no second affection distribution;
- importance of data itself.

3.Data elements for computation of external metrics are designed to use externally accessible information, because it is helpful for end users, operators, maintainers or acquirers to use external metrics. Therefore, counting events and times used here are different from corresponding internal metric.

4. It is recommended that penetration tests be performed to simulate attack, because such security attacks do not normally occur in the usual testing.
Real security metrics may only be taken in "real life system environment", that is "quality in use"

5. This metric is suggested as an experimental use.

6. Data backup is one of the effective ways to prevent data corruption. The creation of back up ensures that necessary data can be restored quickly in the event that parts of the operative date are lost. However, data back up is regarded as a part of the composition of the reliability metrics in this report.

7. It is suggested that this metric be used experimentally.

Table 8.1.5 Functionality compliance metrics

External functionality compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional compliance	How compliant is the functionality of the product to applicable regulations, standards and conventions?	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance in the specification. Design test cases in accordance with compliance items. Conduct functional testing for these test cases. Count the number of compliance items that have been satisfied.	X = 1 - A / B A= Number of functionality compliance items specified that have not been implemented during testing B= Total number of functionality compliance items specified	0<= X <=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Product description (User manual or Specification) of compliance and related standards, conventions or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User
***NOTE**: 1. It may be useful to collect several measured values along time, to analyse the trend 2. It is suggested to count number of failures, because problem detection is an objective of of increasingly satisfied compliance items and to determine whether they are fully satisfied or effective testing and also suitable for counting and recording. not.*									
Interface standard compliance	How compliant are the interfaces to applicable regulations, standards and conventions?	Count the number of interfaces that meet required compliance and compare with the number of interfaces requiring compliance as in the specifications. ***NOTE**: All specified attributes of a standard must be tested.*	X= A / B A= Number of correctly implemented interfaces as specified B= Total number of interfaces requiring compliance	0<=X<= 1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Product description of compliance and related standards, conventions or regulations Test specification and report	6.5 Validation	Developer

Table 8.2.1 Maturity metrics

External maturity metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Estimated latent fault density	How many problems still exist that may emerge as future faults?	Count the number of faults detected during a defined trial period and predict potential number of future faults using a reliability growth estimation model.	X= {ABS(A1 - A2)} / B (X: Estimated residuary latent fault density) ABS()= Absolute Value A1 = total number of predicted latent faults in a software product A2 = total number of actually detected faults B= product size	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	A1= Count A2= Count B= Size X= Count/ Size	Test report Operation report Problem report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation 6.3 Quality Assurance	Developer Tester SQA User

***NOTE**: 1.When total number of actually detected faults becomes larger than total number of predicted latent faults, it is recommended to predict again and estimate more larger number. Estimated larger numbers are intended to predict reasonable latent failures, but not to make the product look better.*

2.It is recommended to use several reliability growth estimation models and choose the most suitable one and repeat prediction with monitoring detected faults.

3. It may be helpful to predict upper and lower number of latent faults.

4. It is necessary to convert this value (X) to the <0,1> interval if making summarisation of characteristics

External maturity metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

External maturity metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Failure density against test cases	How many failures were detected during defined trial period?	Count the number of detected failures and performed test cases.	X= A1 / A2 A1 = number of detected failures A2 = number of performed test cases	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	A1= Count A2= Count B = Size X,Y= Count/ Size	Test report Operation report Problem report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.3 Quality Assurance	Developer Tester SQA

***NOTE**: 1. The larger is the better, in early stage of testing. On the contrary, the smaller is the better, in later stage of testing or operation. It is recommended to monitor the trend of this measure along with the time.*
2. This metric depends on adequacy of test cases so highly that they should be designed to include appropriate cases: e.g., normal, exceptional and abnormal cases.
3. It is necessary to convert this value (X) to the <0,1> interval if making summarisation of characteristics.

Failure resolution	How many failure conditions are resolved?	Count the number of failures that did not reoccur during defined trial period under the similar conditions. Maintain a problem resolution report describing status of all the failures.	X= A1 / A2 A1 = number of resolved failures A2 = total number of actually detected failures	0<=X<= 1 The closer to 1.0 is better as more failures are resolved.	a) Absolute	A1= Count A2= Count A3 = Count X= Count/ Count	Test report Operation (test) report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation	User SQA Maintainer

***NOTE**:*
1. It is recommended to monitor the trend when using this measure.
2. Total number of predicted latent failures might be estimated using reliability growth models adjusted with actual historical data relating to similar software product. In such a case, the number of actual and predicted failures can be comparable and the number of residual unresolved failures can be measurable.

External maturity metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Fault density	How many faults were detected during defined trial period?	Count the number of detected faults and compute density.	X= A / B A = number of detected faults B = product size	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	A= Count B = Size X= Count/ Size	Test report Operation report Problem report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.3 Quality Assurance	Developer Tester SQA

***NOTE**: 1. The larger is the better, in early stage of testing. On the contrary, the smaller is the better, in later stage of testing or operation. It is recommended to monitor the trend of this measure along with the time.*
2. The number of detected faults divided by the number of test cases indicates effectiveness of test cases.
3. It is necessary to convert this value (X) to the <0,1> interval if making summarisation of characteristics.
4. When counting faults, pay attention to the followings:
- Possibility of duplication, because multiple reports may contain the same faults as other report;
- Possibility of others than faults, because users or testers may not figure out whether their problems are operation error, environmental error or software failure.

External maturity metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computetions	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Fault removal	How many faults have been corrected?	Count the number of faults removed during testing and compare with the total number of faults detected and total number of faults predicted.	a) X= A1 / A2 A1 = number of corrected faults A2 = total number of actually detected faults b) Y= A1 / A3 A3 = total number of predicted latent faults in the software product	0<=X<= 1 The closer to 1.0 is better as fewer faults remain. 0<=Y The closer to 1.0 is better as fewer faults remain.	a) Absolute b) Absolute	A1= Count A2= Count A3= Count X= Count/ Count Y= Count/ Count	Test report Organizati on database	5.3 Integration 5.3 Qualifica-tion testing 6.5 Validation 6.3 Quality Assurance	Developer SQA Maintainer

NOTE:

1. It is recommended to monitor the trend during a defined period of time.

2. Total number of predicted latent faults may be estimated using reliability growth models adjusted with actual historical data relating to similar software product.

3. It is recommended to monitor the estimated faults resolution ratio Y, so that if Y > 1, investigate the reason whether it is because more faults have been detected early or because software product contains an unusual number of faults.

Otherwise, when Y < 1, investigate whether it is because there are less than the usual number of defects in the software products or because the testing was not adequate to detect all possible faults.

4. It is necessary to convert this value (Y) to the <0,1> interval if making summarisation of characteristics

5. When counting faults, pay attention to the possibility of duplication, because multiple reports may contain the same faults as other report.

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computetions	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Mean time between failures (MTBF)	How frequently does the software fail in operation?	Count the number of failures occurred during a defined period of operation and compute the average interval between the failures.	a) X = T1 / A b) Y = T2 / A T1 = operation time T2 = sum of time intervals between consecutive failure occurrences A = total number of actually detected failures (Failures occurred during observed operation time)	0<X,Y The longer is the better.As longer time can be expected between failures.	a) Ratio b) Ratio	A = Count T1 = Time T2 = Time X =Time / Count Y =Time/ Count	Test report Operation (test) report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation testing 5.4 Operation	Maintainer User

NOTE:

1. The following investigation may be helpful: - distribution of time interval between failure occurrences;
- changes of mean time along with interval operation time period;
- distribution indicating which function has frequent failure occurrences and operation because of function and use dependency.

2. Failure rate or hazard rate calculation may be altematively used.

3. It is necessary to convert this value (X,Y) to the <0,1> interval if making summarisation of characteristics

External maturity metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Test coverage (Specified operation scenario testing coverage)	How much of required test cases have been executed during testing?	Count the number of test cases performed during testing and compare the number of test cases required to obtain adequate test coverage.	X= A / B A= Number of actually performed test cases representing operation scenario during testing B= Number of test cases to be performed to cover requirements	0<=X<=1 The closer to 1.0 is the better test coverage.	Absolute	A= Count B= Count X= Count/ Count	Req. spec., Test spec. or User manual Test report Operation report	5.3 Qualifica-tion testing 6.5 Validation 6.3 Quality Assurance	Developer Tester SQA
NOTE: *1. Test cases may be normalised by software size, that is: test density coverage Y= A / C, where C= Size of product to be tested. The larger Y is the better. Size may be functional size that user can measure.*									
Test maturity	Is the product well tested? *(**NOTE**: This is to predict the success rate the product will achieve in future testing.)*	Count the number of passed test cases which have been actually executed and compare it to the total number of test cases to be performed as per requirements.	X= A / B A= Number of passed test cases during testing or operation B= Number of test cases to be performed to cover requirements	0<=X<=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Req. spec., Test spec., or User manual Test report Operation report	5.3 Qualifica-tion testing 6.3 Quality Assurance	Developer Tester SQA
***NOTE**: 1. It is recommended to perform stress testing using live historical data especially from peak periods.* *It is also recommended to ensure that the following test types are executed and passed successfully:* *- User operation scenario;* *- Peak stress;* *- Overloaded data input..*			*2. Passed test cases may be normalised by software size, that is: passed test case density Y= A / C, where C= Size of product to be tested. The larger Y is better. Size may be functional size that user can measure.*						

Table 8.2.2 Fault tolerance metrics

External fault tolerance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Breakdown avoidance	How often the software product causes the break down of the total production environment?	Count the number of breakdowns occurrence with respect to number of failures. If it is under operation, analyse log of user operation history.	X= 1- A / B A= Number of breakdowns B= Number of failures	0<=X<= 1 The closer to 1.0 is the better.	Absolute	A =Count B =Count X =Count/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation	User Maintainer
NOTE: *1.The breakdown means the execution of any user tasks is suspended until system is restarted, or its control is lost until system is forced to be shut down.*				*2. When none or few failures are observed, time between breakdowns may be more suitable.*					
Failure avoidance	How many fault patterns were brought under control to avoid critical and serious failures?	Count the number of avoided fault patterns and compare it to the number of fault patterns to be considered	X=A / B A= Number of avoided critical and serious failure occurrences against test cases of fault pattern B= Number of executed test cases of fault pattern (almost causing failure) during testing	0<=X<= 1 The closer to 1.0 is better, as the user can more often avoid critical or serious failure.	Absolute	A= Count B= Count X= Count/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation	User Maintainer
	NOTE: *1. It is recommended to categorise failure avoidance levels which is the extent of mitigating impact of faults, for example:* *-Critical: entire system stops / or serious database destruction;* *-Serious: important functions become inoperable and no alternative way of operating (workaround);* *-Average: most functions are still available, but limited performance occurs with limited or alternate operation (workaround);* *-Small: a few functions experience limited performance with limited operation;* *-None: impact does not reach end user*			*2. Failure avoidance levels may be based on a risk matrix composed by severity of consequence and frequency of occurrence provided by ISO/IEC 15026 System and software integrity.* *3. Fault pattern examples* *- out of range data* *- deadlock* *Fault tree analysis technique may be used to detect fault patterns.* *4. Test cases can include the human incorrect operation*				

External fault tolerance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Incorrect operation avoidance	How many functions are implemented with incorrect operations avoidance capability?	Count the number of test cases of incorrect operations which were avoided to cause critical and serious failures and compare it to the number of executed test cases of incorrect operation patterns to be considered.	X=A / B A= Number of avoided critical and serious failures occurrences B= Number of executed test cases of incorrect operation patterns (almost causing failure) during testing	0<=X<= 1 The closer to 1.0 is better, as more incorrect user operation is avoided.	Absolute	A= Count B= Count X= Count/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation	User Maintainer

NOTE:
1. Also data damage in addition to system failure.
2. Incorrect operation patterns
- Incorrect data types as parameters
- Incorrect sequence of data input
- Incorrect sequence of operation
3. Fault tree analysis technique may be used to detect incorrect operation patterns
4. This metric may be used experimentally.

Table 8.2.3 Recoverability metrics

External recoverability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Availability	How available is the system for use during the specified period of time?	Test system in a production like environment for a specified period of time performing all user operations. Measure the repair time period each time the system was unavailable during the trial. Compute mean time to repair.	a) X= { To / (To + Tr) } b) Y= A1 / A2 To = operation time Tr = time to repair A1= total available cases of user's successful software use when user attempt to use A2= total number of cases of user's attempt to use the software during observation time. This is from the user callable function operation view.	0<=X<=1 The larger and closer to 1.0 is better, as the user can use the software for more time. 0<=Y<=1 The larger and closer to 1.0 is the better.	(a),(b) Absolute	To = Time Tr = Time X= Time/ Time A1= Count A2= Count Y= Count/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation	User Maintainer
***NOTE**: It is recommended that this metric includes only the automatic recovery provided by the software and excludes the maintenance work of human.*									
Mean down time	What is the average time the system stays unavailable when a failure occurs before gradual start up?	Measure the down time each time the system is unavailable during a specified trial period and compute the mean time.	X= T / N T= Total down time N= Number of observed breakdowns The worst case or distribution of down time should be measured.	0<X The smaller is the better, system will be down for shorter time.	Ratio	T= Time N= Count X= Time/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation	User Maintainer

NOTE:

1. It is recommended that this recoverability metric includes only the automatic recovery provided by the software and excludes the maintenance work of human.

2. It is necessary to convert this value (X) to the <0,1> interval if making summarisation of characteristics

External recoverability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Mean recovery time	What is the average time the system takes to complete recovery from initial partial recovery?	Measure the full recovery times for each of the time the system was brought down during the specified trial period and compute the mean time.	X= Sum(T) / B T= Time to recovery downed software system at each opportunity N= Number of cases which observed software system entered into recovery	0<X The smaller is the better.	Ratio	T= Time N= Count X= Time/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation	User Maintainer
NOTE: *1. It is recommended to measure the maximum time of the worst case or distribution of recovery time for many cases.* *2. It is recommended that this recoverability metric includes only the automatic recovery provided by the software and excludes the maintenance work of human.* *3. It is recommended to distinguish the grades of recovery difficulty, for example, recovery of destroyed database is more difficult than recovery of destroyed transaction.* *4. It is necessary to convert this value (X) to the <0,1> interval if making summarisation of characteristics*									
Restartability	How often the system can restart providing service to users within a required time?	Count the number of times the system restarts and provides service to users within a target required time and compare it to the total number of restarts, when the system was brought down during the specified trial period.	X = A / B A= Number of restarts which met to required time during testing or user operation support B= Total number of restarts during testing or user operation support	0<=X<=1 The larger and closer to 1.0 is better, as the user can restart easily.	Absolute	A =Count B =Count X =Count/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation	User Maintainer
NOTE: *1. It is recommended to estimate different time to restart to correspond to the severity level of inoperability, such as data base destruction, lost multi transaction, lost single transaction, or temporary data destruction.* *2. It is recommended that this recoverability metric includes only the automatic recovery provided by the software and excludes the maintenance work of human.*									

External recoverability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Restorability	How capable is the product in restoring itself after abnormal event or at request?	Count the number of successful restorations and compare it to the number of tested restoration required in the specifications. Restoration requirement examples: database checkpoint, transaction checkpoint, redo function, undo function etc.	X= A / B A= Number of restoration cases successfully done B= Number of restoration cases tested as per requirements	0<=X<=1 The larger and closer to 1.0 is better, as he product is more capable to restore in defined cases.	Absolute	A= Count B= Count X= Count/ Count	Req. spec., Test spec. or User manual Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation	User Maintainer
***NOTE**: It is recommended that this metric includes only the automatic recovery provided by the software and excludes the maintenance work of human.*									
Restore effectiveness	How effective is the restoration capability?	Count the number of tested restoration meeting target restoration time and compare it to the number of restorations required with specified target time.	X= A / B A= Number of cases successfully restored meeting the target restore time B= Number of cases performed	0<=X<=1 The larger and closer to 1.0 is the better, as the restoration process in product is more effective.	Absolute	A= Count B= Count X= Count/ Count	Test report Operation report	5.3 Integration 5.3 Qualifica-tion testing 5.4 Operation 6.5 Validation	User Maintainer
***NOTE**: It is recommended that this metric includes only the automatic recovery provided by the software and excludes the maintenance work of human.*									

Table 8.2.4 Reliability compliance metrics

External reliability compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Reliability compliance	How compliant is the reliability of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.	X = 1 - A / B A= Number of reliability compliance items specified that have not been implemented during testing B= Total number of reliability compliance items specified	0<= X <=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Product description (User manual or Specifica-tion) of complian-ce and related standards, conven-tions or regulations Test specifica-tion and report	5.3 Qualifica-tion testing 6.5 Validation	Supplier User

NOTE:
It may be useful to collect several measured values along time, to analyse the trend of increasingly satisfied compliance items and to determine whether they are fully satisfied or not.

Table 8.4.1 Time behaviour metrics a) Response time

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Response time	What is the time taken to complete a specified task? How long does it take before the system response to a specified operation?	Start a specified task. Measure the time it takes for the sample to complete its operation. Keep a record of each attempt.	T = (time of gaining the result) - (time of command entry finished)	0 < T The sooner is the better.	Ratio	T= Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
***NOTE**: It is recommended to take account of time bandwidth and to use statistical analysis with measures for a lot of tasks (sample shots) and not for only one task.*									
Response time (Mean time to response)	What is the average wait time the user experiences after issuing a request until the request is completed within a specified system load in terms of concurrent tasks and system utilisation?	Execute a number of scenarios of concurrent tasks. Measure the time it takes to complete the selected operation(s). Keep a record of each attempt and compute the mean time for each scenario.	X = Tmean / TXmean Tmean = Σ(Ti) / N, (for i=1 to N) TXmean = required mean response time Ti= response time for i-th evaluation (shot) N= number of evaluations (sampled shots)	0 <= X The nearer to 1.0 and less than 1.0 is the better.	Absolute	Tmean= Time TXmean= Time Ti= Time N= Count X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
***NOTE**: Required mean response time can be derived from specification of required real-time processing, user expectation of business needs or observation of user reaction. A user cognitive of the aspects of human ergonomics might be considered.*									

External time behaviour metrics a) Response time										
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Response time (Worst case response time ratio)	What is the absolute limit on time required in fulfilling a function? In the worst case, can user still get response within the specified time limit? In the worst case, can user still get reply from the software within a time short enough to be tolerable for user?	Calibrate the test. Emulate a condition whereby the system reaches a maximum load situation. Run application and monitor result(s)	X= Tmax / Rmax Tmax= MAX(Ti) (for i=1 to N) Rmax = required maximum response time MAX(Ti)= maximum response time among evaluations N= number of evaluations (sampled shots) Ti= response time for i-th evaluation (shot) ***NOTE:*** *1. Distribution may be calculated as illustrated below.* *Statistical maximal ratio Y= Tdev / Rmax* *Tdev = Tmean + K (DEV)* *Tdev is time deviated from mean time to the particular time: e.g. 2 or 3 times of standard deviation.* *K: coefficient (2 or 3)* *DEV=SQRT{ ∑((Ti-Tmean) **2) / (N-1)} (for i=1 to N)* *Tmean = ∑(Ti) / N, (for i=1 to N)* *TXmean = required mean response time*	0 < X The nearer to 1 and less than 1 is the better.	Absolute	Tmax= Time Rmax= Time Ti= Time N= Count X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

Table 8.4.1 Time behaviour metrics b) Throughput External time behaviour metrics b) Throughput

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Throughput	How many tasks can be successfully performed over a given period of time?	Calibrate each task according to the intended priority given. Start several job tasks. Measure the time it takes for the measured task to complete its operation. Keep a record of each attempt.	X = A / T A = number of completed tasks T = observation time period	0 < X The larger is the better.	Ratio	A= Count T= Time X= Count/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Throughput (Mean amount of throughput)	What is the average number of concurrent tasks the system can handle over a set unit of time?	Calibrate each task according to intended priority. Execute a number of concurrent tasks. Measure the time it takes to complete the selected task in the given traffic. Keep a record of each attempt.	X = Xmean / Rmean Xmean = Σ(Xi)/N Rmean = required mean throughput Xi = Ai / Ti Ai = number of concurrent tasks observed over set period of time for i-th evaluation Ti = set period of time for i-th evaluation N = number of evaluations	0 < X The larger is the better.	Absolute	Xmean= Count Rmean= Count Ai= Count Ti= Time Xi= Count/ Time N= Count X = Count/ Count	Testing report Operation report showing elapse time	5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Throughput (Worst case throughput ratio)	What is the absolute limit on the system in terms of the number and handling of concurrent tasks as throughput?	Calibrate the test. Emulate the condition whereby the system reaches a situation of maximum load. Run job tasks concurrently and monitor result(s).	X = Xmax / Rmax Xmax = MAX(Xi) (for i = 1 to N) Rmax = required maximum throughput. MAX(Xi) = maximum number of job tasks among evaluations Xi = Ai / Ti Ai = number of concurrent tasks observed over set period of time for i-th evaluation Ti = set period of time for i-th evaluation N= number of evaluations ***NOTE:*** *1. Distribution may be calculated as illustrated below.* *Statistical maximal ratio Y= Xdev / Xmax* *Xdev = Xmean + K (DEV)* *Xdev is time deviated from mean time to the particular time: e.g. 2 or 3 times of standard deviation.* *K: coefficient (2 or 3)* *DEV=SQRT{ Σ((Xi-Xmean) **2) / (N-1)} (for i=1 to N)* *Xmean = Σ(Xi)/N*	0 < X The larger is the better.	Absolute	Xmax= Count Rmax= Count Ai= Count Ti= Time Xi= Count/ Time N= Count Xdev= Count X = Count/ Count	Testing report Operation report showing elapse time	5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

Table 8.4.1 Time behaviour metrics c) Turnaround time xternal time behaviour metrics c) Turnaround time

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Turnaround time	What is the wait time the user experiences after issuing an instruction to start a group of related tasks and their completion?	Calibrate the test accordingly. Start the job task. Measure the time it takes for the job task to complete its operation. Keep a record of each attempt.	T = Time between user's finishing getting output results and user's finishing request ***NOTE**: It is recommended to take account of time bandwidth and to use statistical analysis with measures for many tasks (sample shots), not only one task (shot).*	0 < T The shorter the better.	Ratio	T= Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Turnaround time (Mean time for turnaround)	What is the average wait time the user experiences after issuing an instruction to start a group of related tasks and their completion within a specified system load in terms of concurrent tasks and system utilisation?	Calibrate the test. Emulate a condition where a load is placed on the system by executing a number of concurrent tasks (sampled shots). Measure the time it takes to complete the selected job task in the given traffic. Keep a record of each attempt.	X = Tmean/TXmean Tmean = Σ(Ti)/N, (for i=1 to N) TXmean = required mean turnaround time Ti = turnaround time for i-th evaluation (shot) N = number of evaluations (sampled shots)	0 < X The shorter is the better.	Absolute	Tmean= Time TXmean= Time Ti= Time N= Count X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Turnaround time (Worst case turnaround time ratio)	What is the absolute limit on time required in fulfilling a job task? In the worst case, how long does it take for software system to perform specified tasks?	Calibrate the test. Emulate a condition where by the system reaches maximum load in terms of tasks performed. Run the selected job task and monitor result(s).	X= Tmax / Rmax Tmax= MAX(Ti) (for i=1 to N) Rmax = required maximum turnaround time MAX(Ti)= maximum turnaround time among evaluations N= number of evaluations (sampled shots) Ti= turnaround time for i-th evaluation (shot) ***NOTE:*** *1. Distribution may be calculated as illustrated below.* *Statistical maximal ratio Y= Tdev / Rmax* *Tdev = Tmean + K (DEV)* *Tdev is time deviated from mean time to the particular time: e.g. 2 or 3 times of standard deviation.* *K: coefficient (2 or 3)* *DEV=SQRT{ ∑((Ti-Tmean) **2) / (N-1)} (for i=1 to N)* *Tmean = ∑(Ti) / N, (for i=1 to N)* *TXmean = required mean turnaround time*	0 < X The nearer to 1.0 and less than 1.0 is the better.	Absolute	X= Time/ Time Tmax = Time Rmax= Time Ti= Time N= Count Tdev = Time	Testing report Operation report showing elapse time	5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Waiting time	What proportion of the time do users spend waiting for the system to respond?	Execute a number of scenarios of concurrent tasks. Measure the time it takes to complete the selected operation(s). Keep a record of each attempt and compute the mean time for each scenario.	X = Ta / Tb Ta = total time spent waiting Tb = task time	0<= X The smaller the better.	Absolute	Ta= Time Tb= Time X= Time/ Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

NOTE: *If the tasks can be partially completed, the Task efficiency metric should be used when making comparisons.*

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

Table 8.4.2 Resource utilisation metrics a) I/O devices resource utilization External resource utilisation metrics a) I/O devices resource utilisation

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
I/O devices utilisation	Is the I/O device utilisation too high, causing inefficiencies?	Execute concurrently a large number of tasks, record I/O device utilisation, and compare with the design objectives.	X = A / B A = time of I/O devices occupied B = specified time which is designed to occupy I/O devices	0 <= X <= 1 The less than and nearer to the 1.0 is the better.	Absolute	A= Time B= Time X= Time/ Time	Testing report Operation report	5.3 Qualifica-tion testing 5.4 Operation Mainte-nance	Developer Maintainer SQA
I/O loading limits	What is the absolute limit on I/O utilisation in fulfilling a function?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run application and monitor result(s).	X = Amax / Rmax Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum I/O messages MAX(Ai) = Maximum number of I/O messages from 1st to i-th evaluation. N= number of evaluations.	0<= X The smaller is the better.	Absolute	Amax = Count Rmax = Count Ai = Count N= Count X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
I/O related errors	How often does the user encounter problems in I/O device related operations?	Calibrate the test conditions. Emulate a condition whereby the system reaches a situation of maximum I/O load. Run the application and record number of errors due to I/O failure and warnings.	X = A / T A = number of warning messages or system failures T = User operating time during user observation	0 <= X The smaller is the better.	Ratio	A = Count T = Time X = Count/ Time	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer SQA
Mean I/O fulfillment ratio	What is the average number of I/O related error messages and failures over a specified length of time and specified utilisation?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to I/O failure and warnings.	X = Amean / Rmean Amean = Σ(Ai)/N Rmean = required mean number of I/O messages Ai = number of I/O error messages for i-th evaluation N = number of evaluations	0<= X The smaller is the better.	Absolute	Amean = Count Rmean = Count Ai = Count N= Count X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
User waiting time of I/O devices utilisation	What is the impact of I/O device utilisation on the user wait times?	Execute concurrently a large amount of tasks and measure the user wait times as a result of I/O device operation.	T = Time spent to wait for finish of I/O devices operation ***NOTE**: It is recommended that the maximal and distributed time are to be investigated for several cases of testing or operating, because the measures are tend to be fluctuated by condition of use.*	0 < T The shorter is the better.	Ratio	T= Time	Testing report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

Table 8.4.2 Resource utilisation metrics b) Memory resource utilization External resource utilisation metrics b) Memory resource utilisation

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Maximum memory utilisation	What is the absolute limit on memory required in fulfilling a function?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run application and monitor result(s)	X = Amax / Rmax Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum memory related error messages MAX(Ai) = Maximum number of memory related error messages from 1st to i-th evaluation N= number of evaluations	0<= X The smaller is the better.	Absolute	Amax= Count Rmax= Count Ai= Count N= Count X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Mean occurrence of memory error	What is the average number of memory related error messages and failures over a specified length of time and a specified load on the system?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings.	X = Amean / Rmean Amean = ∑(Ai)/N Rmean = required mean number of memory related error messages Ai = number of memory related error messages for i-th evaluation N = number of evaluations	0<= X The smaller is the better.	Absolute	Amean= Count Rmean= Count Ai= Count N= Count X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Ratio of memory error/time	How many memory errors were experienced over a set period of time and specified resource utilisation?	Calibrate the test conditions. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings.	X = A / T A = number of warning messages or system failures T = User operating time during user observation	0 <= X The smaller is the better.	Ratio	A = Count T = Time X = Count/ Time	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience

Table 8.4.2 Resource utilisation metrics c) Transmission resource utilization External resource utilisation metrics c) Transmission resource utilisation

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Maximum transmission utilisation	What is the absolute limit of transmissions required to fulfil a function?	Evaluate what is required for the system to reach a situation of maximum load. Emulate this condition. Run application and monitor result(s) .	X = Amax / Rmax Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum number of transmission related error messages and failures MAX(Ai) = Maximum number of transmission related error messages and failures from 1st to i-th evaluation. N= number of evaluations	0<= X The smaller is the better.	Absolute	Amax = Count Rmax = Count Ai = Count N= Count X = Count/ Count	Testing report Operation raport showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Media device utilisation balancing	What is the degree of synchronisation between different media over a set period of time?	Calibrate the test conditions. Emulate a condition whereby the system reaches a situation of maximum transmission load. Run the application and record the delay in the processing of different media types.	X = SyncTime/T SyncTime = Time devoted to a continuous resource T = required time period during which dissimilar media are expected to finish their tasks with synchronisation	The smaller is the better.	Ratio	SyncTime = Time T = Time X = Time/Time	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer SQA

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Mean occurrence of transmission error	What is the average number of transmission related error messages and failures over a specified length of time and specified utilisation?	Calibrate the test condition. Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to transmission failure and warnings.	X = Amean / Rmean Amean = ∑(Ai)/N Rmean = required mean number of transmission related error messages and failures Ai = Number of transmission related error messages and failures for i-th evaluation N = number of evaluations	0<= X The smaller is the better.	Absolute	Amean= Count Rmean= Count Ai= Count N= Count X = Count/ Count	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA
Mean transmission error per time	How many transmission -related error messages were experienced over a set period of time and specified resource utilisation?	Calibrate the test conditions. Emulate a condition whereby the system reaches a situation of maximum transmission load. Run the application and record number of errors due to transmission failure and warnings.	X = A / T A = number of warning messages or system failures T = User operating time during user observation	0 <= X The smaller is the better.	Ratio	A = Count T = Time X = Count/ Time	Testing report Operation report showing elapse time	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer SQA
Transmission capacity utilisation	Is software system capable of performing tasks within expected transmission capacity?	Execute concurrently specified tasks with multiple users, observe transmission capacity and compare specified one.	X = A / B A = transmission capacity B = specified transmission capacity which is designed to be used by the software during execution ***NOTE**: It is recommended to measure dynamically peaked value with multiple users.*	0 <= X <= 1 The less than and nearer to the 1.0 is the better.	Absolute	A= Size B= Size X= Size / Size	Testing report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer SQA

Table 8.4.3 Efficiency compliance metrics

Efficiency compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Efficiency Compliance	How compliant is the efficiency of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance in the specification.	X = 1 - A / B (X: Ratio of satisfied compliance items relating to efficiency) A= Number of efficiency compliance items specified that have not been implemented during testing B= Total number of efficiency compliance items specified ***NOTE**: It may be useful to collect several measured values along time, to analyse the trend of increasing satisfied compliance items and to determine whether they are fully satisfied or not.*	0<= X <=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Product description (User manual or Specification) of compliance and related standards, conventions or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User

Table 8.5.2 Changeability metrics ***External changeability metrics***

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Change cycle efficiency	Can the user's problem be solved to his satisfaction within an acceptable time scale?	Monitor interaction between user and supplier. Record the time taken from the initial user's request to the resolution of problem.	Average Time : Tav = Sum(Tu) / N Tu= Trc - Tsn Tsn= Time at which user finished to send request for maintenance to supplier with problem report Trc= Time at which user received the revised version release (or status report) N= Number of revised versions	0<Tav The shorter is the better., except of the number of revised versions was large.	Ratio	Tu= Time Trc, Tsn = Time N= Count Tav= Time	Problem resolution report Mainten-ance report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	User Maintainer Operator
Change implementation elapse time	Can the maintainer easily change the software to resolve the failure problem?	Observe the behaviour of the user and maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	Average Time : Tav = Sum(Tm) / N Tm=Tout - Tin Tout= Time at which the causes of failure are removed with changing the software (or status is reported back to user) Tin= Time at which the causes of failures are found out N= Number of registered and removed failures	0<Tav The shorter is the better, except of the number of failures was large.	Ratio	Tm= Time Tin, Tout = Time Tav= Time	Problem resolution report Mainten-ance report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator

***NOTE**: 1. It is recommended to measure maximal time of the worst case and time bandwidth to represent deviation. 2. It is recommended to exclude the failures for which causes have not yet been found when the measurement is done. However, the ratio of such obscure failures should be also measured and presented together. 3. From the individual user's point of view, time is of concern, while effort may also be of concern from the maintainer's point of view. Therefore, person-hours may be used instead of time.*

Modification complexity	Can the maintainer easily change the software to resolve problem?	Observe behaviour of maintainer who is trying to change the software. Otherwise, investigate problem resolution report or maintenance report and product description.	T = Sum (A / B) / N A= Work time spent to change B= Size of software change N= Number of changes ***NOTE:*** *A size of software change may be changed executable statements of program code, number of changed items of requirements specification, or changed pages of document etc.*	0<T The shorter is the better or the required number of changes were excessive.	Ratio	A= Time B= Size N= Count T= Time	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator
Parameterised modifiability	Can the user or the maintainer easily change parameter to change software and resolve problems?	Observe behaviour of the user or the maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	X=1- A / B A= Number of cases which maintainer fails to change software by using parameter B= Number of cases which maintainer attempts to change software by using parameter	0<=X<= 1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator User
Software change control capability	Can the user easily identify revised versions? Can the maintainer easily change the software to resolve problems?	Observe the behaviour of user or maintainer while trying to change the software. Otherwise, investigate problem resolution report or maintenance report.	X= A / B A= Number of change log data actually recorded B= Number of change log data planned to be recorded enough to trace software changes	0<=X<= 1 The closer to 1.0 is the better or the closer to 0 the fewer changes have taken place.	Absolute	A= Count B= Count X= Count/ Count	User manual or specification Problem resolution report Maintenance report Operation report	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Developer Maintainer Operator

Table 8.5.3 Stability metrics

External stability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Change success ratio	Can user operate software system without failures after maintenance? Can maintainer easily mitigate failures caused by maintenance side effects?	Observe behaviour of user or maintainer who is operating software system after maintenance. Count failures which user or maintainer encountered during operating software before and after maintenance. Otherwise, investigate problem resolution report, operation report or maintenance report.	X= Na / Ta Y = { (Na / Ta) / (Nb / Tb) } Na = Number of cases which user encounters failures during operation after software was changed Nb = Number of cases which user encounters failures during operation before software is changed Ta = Operation time during specified observation period after software is changed Tb = Operation time during specified observation period before software is changed	0<=X,Y The smaller and closer to 0 is the better.	Ratio	Na, Nb= Count Ta,Tb= Time X= Count/ Time Y=[(Count/ Time) / (Count/ Time)]	Problem resolution report Mainten-ance report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator

***NOTE**: 1. X and Y imply " frequency of encountering failures after change" and "fluctuated 3. If changed function is identified, it is recommended to determine whether encountered frequency of encountering failures before/after change". failures are detected in the changed function itself or in the other ones. The extent of impacts 2. User may need specified period to determine side effects of software changes, when the may be rated for each failure. revision-up of software is introduced for resolving problems.*
3. It is recommend to compare this frequency before and after change.

Modification impact localisation (Emerging failure after change)	Can user operate software system without failures after maintenance? Can maintainer easily mitigate failures caused by maintenance side effects?	Count failures occurrences after change, which are mutually chaining and affected by change.	X= A / N A= Number of failures emerged after failure is resolved by change during specified period N= Number of resolved failures	0<=X The smaller and closer to 0 is the better.	Absolute	A= Count N= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator

***NOTE**: X implies "chaining failure emerging per resolved failure". It is recommend to give precise measure by checking whether cause of current failure is attributed to change for previous failure resolution, as possible.*

Table 8.5.4 Testability metrics

External testability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Availability of built-in test function	Can user and maintainer easily perform operational testing without additional test facility preparation?	Observe behaviour of user or maintainer who is testing software system after maintenance.	X= A / B A= Number of cases in which maintainer can use suitably built-in test function B= Number of cases of test opportunities	0 <= X <=1 The larger and the closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator
***NOTE**: Examples of built-in test functions include simulation function, pre-check function for ready to use, etc.*									
Re-test efficiency	Can user and maintainer easily perform operational testing and determine whether the software is ready for operation or not?	Observe behaviour of user or maintainer who is testing software system after maintenance.	X= Sum(T) / N T= Time spent to test to make sure whether reported failure was resolved or not N= Number of resolved failures	0<X The smaller is the better.	Ratio	T= Time N= Count X= Time /Count	Problem resolution report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator
NOTE: X implies "average time (effort) to test after failure resolution". If failures are not resolved or fixed, exclude them and separately measure ratio of such failures.									
Test restartability	Can user and maintainer easily perform operational testing with check points after maintenance?	Observe behaviour of user or maintainer who is testing software system after maintenance.	X = A / B A = Number of cases in which maintainer can pause and restart executing test run at desired points to check step by step B= Number of cases of pause of executing test run	0 <= X <=1 The larger and the closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Problem resolution report Operation report	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Developer Maintainer Operator

Table 8.5.5 Maintainability compliance metrics

External maintainability compliance metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Target audience
Maintainability compliance	How compliant is the maintainability of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance in the specification.	X = 1- A / B A= Number of maintainability compliance items specified that have not been implemented during testing B= Total number of maintainability compliance items specified	0<= X <=1 The closer to 1.0 is the better.	Absolute	A= Count B= Count X= Count/ Count	Product description (User manual or Specification) of compliance and related standards, conventions or regulations Test specification and report	5.3 Qualification testing 6.5 Validation	Supplier User

NOTE:
It may be useful to collect several measured values along time, to analyse the trend of increasing satisfied compliance items and to determine whether they are fully satisfied.