UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Centro de Informatización de la Gestión de Entidades

MÉTODO PARA LA EVALUACIÓN ARQUITECTÓNICA DE CEDRUX

Trabajo final presentado en opción al título de
Máster en Informática Aplicada

Autora: Ing. Larisa González Alvarez
Tutores: Dr. Oscar Antonio González Chong
MSc. Osmar Leyet Fernández
MSc. Michaell González Jorrín

La Habana
2012

A mi familia, cuyo aporte está en cada página de este trabajo.

A mis amigos, por el soporte emocional que me mantuvo con fuerzas.

A Donald Firesmith, por sus recomendaciones certeras y su ayuda desinteresada y oportuna.

A mis tutores, por sus revisiones y opiniones.

A todos los que han aportado ideas valiosas para el desarrollo de esta investigación.

A los que han sido mi apoyo en las largas horas de trabajo.

A mi familia, el pilar fundamental de todas las obras de mi vida.

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo **Larisa González Alvarez**, con carné de identidad **85030901617**, soy el autor principal del trabajo final de maestría **Método para la evaluación arquitectónica de Cedrux**. Finalmente asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ___ días del mes de _____ del año _____.

_____ _____

Ing. Larisa González Alvarez Dr. Oscar González Chong

_____ _____

MSc. Osmar Leyet Fernández MSc. Michaell González Jorrín

RESUMEN

El papel de un método de evaluación de arquitectura, como vía para el aumento de la capacidad de prueba de la calidad interna del software, es el objetivo de esta investigación. Mediante la misma, se detectan oportunidades de mejora en los métodos existentes hasta el momento, referidas al análisis de la calidad integral de sistemas y al análisis de conflictos y balances entre atributos de calidad. Para esto se realiza una propuesta de variación a QUASAR[1], basada en el uso de estadígrafos como la Media aritmética, la Varianza y la Desviación estándar, para la determinación de las prioridades de calidad de un software. Además, se hace uso de algoritmos clásicos de programación como los Vuelta atrás, para el análisis de relaciones entre atributos. Con la propuesta se logra favorecer la capacidad de prueba para este tipo de sistemas y se dota al método QUASAR de nuevos instrumentos de soporte para llevar a cabo el proceso de evaluación de la arquitectura. De esta manera se fortalece dicho método para su aplicabilidad bajo las condiciones actuales del desarrollo de sistemas modulares integrados en la Universidad de las Ciencias Informáticas, tomando como caso de estudio el desarrollo de Cedrux[2].

Palabras claves: atributos de calidad, evaluación de arquitectura, métodos de evaluación de arquitectura

ABSTRACT

The role of architecture evaluation methods as a way of favoring the testability of internal software quality is the objective of this research. By this means, opportunities are found to improve existing methods so far, such as the comprehensive quality testing to ensure architectural integrity across multiple subsystems, and system conflicts and tradeoffs between quality attributes. For this, a variation of QUASAR is proposed, based on the use of statisticians such as the Arithmetic Mean, the Variance and Standard Deviation as quantitative methods for determining the priorities of software quality. It´s also proposed to use a Backtracking algorithm for the analysis of quality attributes relationships. This proposal promotes internal quality checks of such systems and gives QUASAR new support instruments to carry out the architecture evaluation process. This will strengthen the applicability of this method under the current conditions of the development of integrated modular systems at the University of Information Sciences, taking as a case study Cedrux development.

Keywords: quality attributes, architecture assessment, architecture evaluation method

[1] A Method for the QUality Assessment of Software-Intensive System Architectures [4]
[2] Sistema de planificación de recursos empresariales

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los sistemas de Planificación de Recursos Empresariales (ERP[3], por sus siglas en inglés) han devenido en una potente herramienta de gestión integrada que ha permitido superar todos los sistemas de información y gestión que le precedieron. Entre sus múltiples ventajas están [1]:

1. Automatizan y simplifican procesos que se realizan de forma manual, con los consiguientes ahorros de tiempo de operación, mejoramiento de la productividad y aumento de la competitividad de la empresa.
2. Integran todas las áreas de una organización de manera que ésta tiene más control sobre su operación, estableciendo lazos de cooperación y coordinación entre los distintos departamentos, facilitando el proceso de control y auditoría.
3. Permiten disponer de una solución integrada para algunas de las funciones de la organización, lo cual garantiza la actualización continua e inmediata de los datos en las diversas zonas geográficas donde se ubique la organización, mejorando así el proceso de la toma de decisiones.

Los ERP son sistemas informáticos únicos que permiten la completa integración de todos los flujos de información generados desde cualquier área funcional de la empresa almacenándolos en una única base de datos [2].

Unido a la creciente demanda empresarial de este tipo de sistemas, surgen diversas polémicas y retos para los informáticos en su construcción. Es imprescindible lograr un diseño del sistema que permita garantizar las características básicas de un ERP [3]:

- ✓ *Integrales*. Deben permitir controlar los diferentes procesos de la organización, entendiendo que todos los departamentos de una empresa se relacionan entre sí.
- ✓ *Modulares.* Una empresa está formada por un conjunto de departamentos que se encuentran interrelacionados por la información que comparten y que se genera a partir de sus procesos. Los ERP, tanto económica como técnicamente, deben tener dividida su funcionalidad en módulos, los cuales pueden instalarse de acuerdo con los requisitos del cliente.
- ✓ *Adaptables*. Deben estar creados para adaptarse a la forma de trabajo de cada empresa. Esto se logra por medio de la configuración o parametrización de los procesos de acuerdo con los requisitos específicos de la empresa. La parametrización es el valor añadido fundamental que se debe hacer con cualquier ERP para adaptarlo a las necesidades concretas de cada empresa.

[3] Enterprise Resources Planning

Estas particularidades hacen que las soluciones arquitectónicas dadas a los sistemas ERP suelan ser complejas y voluminosas, mucho más cuando se persigue tomar las mejores decisiones de diseño para la construcción de sistemas no solo funcionales sino también rápidos, seguros, escalables, usables. El logro de tales cualidades marca la diferencia entre los productos que se encuentran en el mercado y es por ello que los equipos de desarrollo de software centran su atención en la garantía de este valor agregado.

"El diseño de la arquitectura de un sistema es el proceso por el cual se define una solución para los requisitos técnicos y operacionales del mismo" [4].

Este proceso define qué componentes forman parte del sistema, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada cumpliendo con los criterios de calidad indicados como seguridad, disponibilidad, eficiencia o usabilidad [5].

Cuba, con el fin de elevar la eficiencia y control de sus procesos empresariales, se encuentra inmersa en la obtención de un sistema ERP denominado Cedrux. Durante su desarrollo, se han evidenciado una serie de inconvenientes y riesgos indeseados, tales como:

1. Atrasos en el desarrollo del producto por no lograr el mínimo de calidad requerido, habiéndose visto afectada la funcionalidad de la aplicación.
2. Las soluciones ante inoperatividad del sistema han resultado muy costosas en tiempo y esfuerzo dedicado; en ocasiones han significado rediseño total de la solución.
3. Las formas de detección y corrección de los errores en las decisiones arquitectónicas tomadas, han sido complejas y se ha sobrecargado al programador con estas tareas.

La ocurrencia de estos inconvenientes denota problemas en la capacidad de prueba que en estos momentos se tiene en el proceso de construcción de la arquitectura de Cedrux.

Para efectos de la presente investigación, la capacidad de prueba se entiende como: "la capacidad del producto del software de ser objeto de un diagnóstico para detectar deficiencias o causas de los fallos totales en el software, o para identificar las partes que van a ser modificadas" [16].

Con el fin de identificar aquellos elementos que propician la ocurrencia de tales inconvenientes, se realizó una encuesta a 9 arquitectos del Proyecto ERP-Cuba que han estado a cargo de las soluciones arquitectónicas de varios subsistemas de Cedrux. Los factores identificados son:

1. *Deficiencias en el levantamiento y la documentación de los requisitos no funcionales*

(RNF) del sistema, los cuales constituyen la cantera de atributos de calidad[4] *del software*.

2. *Pobre especificación de los criterios de calidad y su prioridad a seguir para la construcción de la arquitectura*: el 67% de los encuestados no utiliza ningún modelo de calidad como base para la definición, desglose y caracterización de la calidad del producto y no tuvo en cuenta los efectos de la priorización de un atributo sobre otro.
3. *Utilización intuitiva de los objetivos de calidad y sus conflictos, por los arquitectos y desarrolladores en la construcción aislada de los módulos*: solo el 22% de los encuestados se guía por los requisitos no funcionales para establecer la prioridad de sus metas de calidad y un 33% reconoce realizar este paso intuitivamente.
4. *Insuficiente verificación de las metas de calidad del software obtenido, fundamentalmente en su interacción modular integral*: solo el 33% de los encuestados ha realizado evaluaciones a su arquitectura y solo el 11% aplica algún método para la evaluación arquitectónica.
5. *Pobre definición y especificación de responsables, tareas, artefactos y momentos de evaluación arquitectónica dentro del flujo de desarrollo*: solo el 33% de los encuestados ha realizado evaluaciones a su arquitectura y esto se ha llevado a cabo en la fase de Implementación.

Todos estos elementos denotan la necesidad de aplicación de técnicas ingenieriles que permitan conjugar las buenas prácticas de construcción de arquitecturas con las de evaluación de la calidad arquitectónica, de manera que se tribute a una mayor satisfacción de los elementos no funcionales y la calidad subsecuente del producto final.

Problema

¿Cómo desarrollar el proceso de evaluación de la arquitectura de software de Cedrux, de manera que se aumente la capacidad de prueba de la calidad interna del sistema?

Objeto de estudio

El proceso de evaluación de arquitectura de software.

Objetivo

Obtener un método para desarrollar el proceso de evaluación de la arquitectura de Cedrux que aumente la capacidad de prueba de la calidad interna del sistema.

Objetivos específicos

1. Resumir el estado actual de la evaluación de arquitecturas de software, con énfasis en

[4] También conocidos como factores, características u objetivos. Se definen como las propiedades de un servicio que presta el sistema a sus usuarios, constituyen también criterios de medida de la calidad del software.

los métodos de evaluación, para sentar las bases teóricas de la investigación.

2. Obtener un método de evaluación de arquitectura que favorezca la capacidad de prueba de la calidad interna en los sistemas modulares integrados.
3. Evaluar la aplicabilidad del método en el favorecimiento de la capacidad de prueba de la calidad interna de Cedrux.

Campo de acción

Métodos de evaluación de arquitecturas de software.

Tareas de investigación

1. Sintetizar el estudio del estado del arte en materia de evaluación de arquitectura de software para resumir su estado actual.
2. Caracterizar las técnicas y métodos de evaluación que puedan ser utilizados en escenarios arquitectónicos presentes en sistemas de planificación de recursos empresariales.
3. Procesar y evaluar la información obtenida de la investigación del tema y adoptar una posición.
4. Definir un método de base para la evaluación arquitectónica de Cedrux.
5. Adaptar el método seleccionado al proceso de desarrollo de software en curso.
6. Aplicar el método en el sistema Cedrux para determinar la capacidad de prueba de la calidad interna del software.
7. Evaluar el cumplimiento de las variables de la investigación con la aplicación del método y el impacto producido en la capacidad de prueba de la calidad interna del sistema.

Hipótesis

Si se emplea un método de evaluación, aplicable en el contexto de Cedrux, para desarrollar el proceso de evaluación de su arquitectura, se aumentará la capacidad de prueba de la calidad interna del sistema.

Métodos

Para el desarrollo de la investigación se hizo uso de los siguientes métodos científicos:

Histórico-lógico: Para el análisis de la esencia, trayectoria, evolución y estado actual de las evaluaciones de software y de los sistemas de Planificación de Recursos Empresariales a nivel mundial y en Cuba.

Hipotético-deductivo: Para solucionar el problema planteado se tomaron como partida reglas generales de desarrollo y evaluación de arquitectura de software, las mismas fueron evolucionadas y particularizadas a partir de deducciones lógicas para la obtención de nuevo

conocimiento que luego fue verificado empíricamente.

Sistémico: Para analizar el impacto de los elementos que conforman la calidad arquitectónica de un sistema de Planificación de Recursos Empresariales, y sus relaciones.

Observación: Para la identificación de las manifestaciones del problema en el entorno real, de manera que permita caracterizarlo en una vista externa del mismo.

Encuesta: Para el reconocimiento documentado de las variables de calidad presentes en los sistemas del campo de acción y el estado de su análisis, según la perspectiva de los arquitectos involucrados en la creación de dichos sistemas.

Medición: Para cuantificar las unidades de los indicadores en el entorno de desarrollo de Cedrux.

Experimentación: Para la puesta en práctica del método y la obtención de los resultados tras su aplicación en el contexto de desarrollo de Cedrux.

Aporte

1. Método para desarrollar el proceso de evaluación de la arquitectura de Cedrux.
2. Inclusión en el proceso de evaluación de la calidad de la arquitectura de:
   - ✓ Análisis cuantitativo de conflictos y priorización entre atributos de calidad.
   - ✓ Análisis de calidad de la integración.
3. Aumento de la capacidad de prueba de la calidad interna del sistema.

Estructura del documento

El presente trabajo está conformado por tres capítulos.

Capítulo 1: Este capítulo contiene un análisis de la evaluación arquitectónica a través de la presentación de sus conceptos fundamentales y su relación con los sistemas ERP. Asimismo, somete a juicio la adecuación de los métodos de evaluación existentes para el contexto arquitectónico de Cedrux.

Capítulo 2: En este capítulo se presenta la propuesta de solución al problema de la investigación, mediante la adaptación y mejora de QUASAR para la evaluación de la arquitectura de Cedrux. Las variaciones planteadas se centran en la verificación de la calidad integral de la arquitectura de sistemas modulares y el tratamiento de relaciones entre atributos de calidad en la evaluación definida por QUASAR. Para la solución trazada se hace uso de métodos estadísticos y algoritmos clásicos sobre grafos.

Capítulo 3: Este capítulo se refiere a la validación de la propuesta, haciéndose uso para ello de la experimentación y del estudio de casos. Para el experimento se emplean los Test Mann Whitney y Wilcoxon y para la verificación de validez del algoritmo de grafos se aplica el mismo en un ambiente de desarrollo de Cedrux.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En aras de establecer un dominio teórico de los elementos fundamentales que soportan la propuesta de solución, se relacionan en este capítulo los principales conceptos asociados a la evaluación de arquitecturas y su papel en el logro de la calidad de un producto de software.

1.1. Calidad y arquitectura de software

Varios autores ([6], [7], [8], [9]), coinciden en que en gran medida el logro de la calidad del producto de software recae sobre la construcción de la arquitectura, de ahí que se evalúe esta periódicamente para determinar la calidad que atribuye al sistema.

Todo el proceso de evaluación está marcado entonces por la siguiente interrogante: ¿Qué significa calidad? ¿Cómo se identifica?

1.1.1. Modelos de calidad del producto

Hoffman plantea que: *"Para garantizar que un producto de software se corresponda con la calidad exigida se han desarrollado los llamados modelos de calidad. Estos modelos se basan en la descomposición de la calidad total en características abstractas de calidad que son necesarias para satisfacer las necesidades de las partes interesadas"* [10].

Es importante tener en cuenta que las partes interesadas[5] toman varias dimensiones, dígase: arquitectos, desarrolladores del producto, clientes, usuarios. De esto se deduce que "solo pocos atributos de calidad son mencionados en la especificación de requisitos" [11].

Modelo de McCall

McCall es considerado por algunos autores como el primero y más usado de los modelos de calidad del producto [11].

Este modelo es útil para el enfoque integrado de la calidad. En el mismo, los atributos de calidad de software se clasifican en una jerarquía de tres niveles tal como se muestra en la figura 1 [12].

[5] stakeholders, del inglés

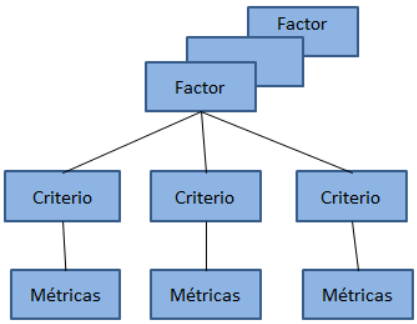


Figura 1. Niveles del modelo de McCall [12]

Según este modelo, un "factor de calidad" representa una característica de comportamiento del sistema, como se ve por los usuarios. A su vez, se conocen como "criterios de calidad a los atributos de un factor de calidad que se relacionan con la producción de software y diseño, desde el punto de vista de los desarrolladores. Una métrica de calidad es una medida que refleja algún aspecto de un criterio de calidad [13].

Modelo Boehm

El modelo de Boehm toma como punto de partida el modelo de McCall, sin embargo le añade 19 criterios, incluyendo características de rendimiento de hardware que no contempla McCall [11]. El árbol que contempla el desglose de características de calidad propuesto por Boehm se muestra en la figura 2.

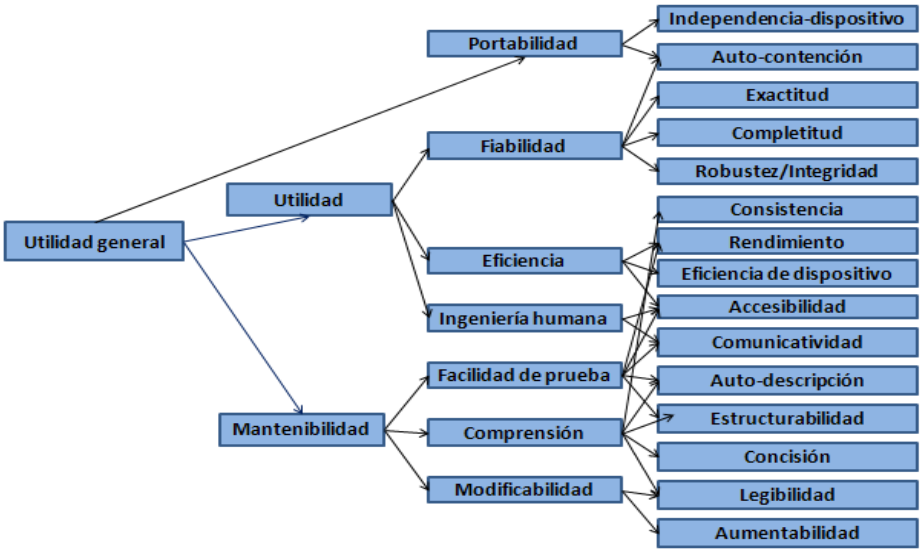


Figura 2. Árbol de características de calidad de Boehm [11]

Modelo ISO/IEC 9126

En el año 1991 la ISO (International Organization for Standardization) publicó su modelo de calidad para la evaluación del producto de software (ISO 9126:1991), que fue extendiendo con revisiones hasta 2004, dando lugar a la actual norma ISO/IEC 9126 "Software Engineering. Product Quality" [14].

El estándar ISO-IEC 9126-1 propone un modelo jerárquico compuesto por características y subcaracterísticas (también conocidos como atributos y subatributos) en términos de calidad interna, calidad externa y calidad en el uso. Muchos investigadores usan este modelo para evaluar paquetes de software [15].

La Oficina Nacional de Normalización de Cuba ha dispuesto una adaptación de esta Norma para Cuba donde tipifica los atributos de calidad del software, como se muestra en la figura 3.

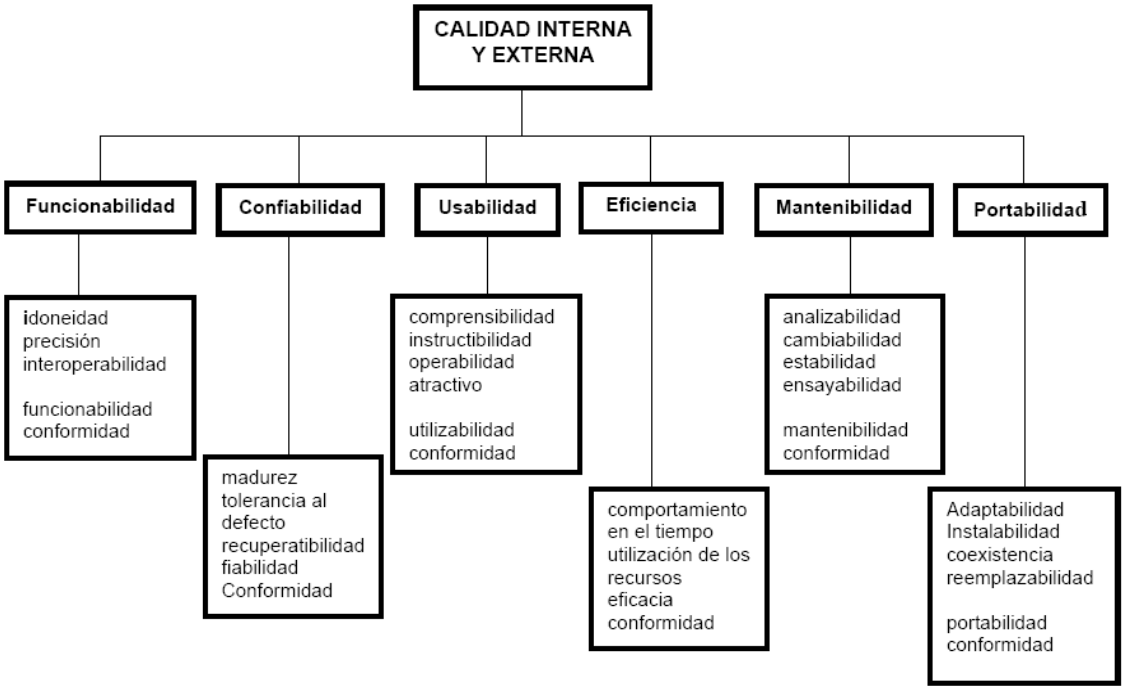


Figura 3. Atributos de calidad del software según ISO/IEC 9126-1 [16]

La norma de calidad ISO/IEC 9126-1 cuenta a su favor:

1. Es una normativa respaldada por una organización reconocida internacionalmente en la temática de estandarización de calidad de productos y con un alto prestigio en ese campo.
2. Ha sido acogida por Cuba como norma regulatoria en cuanto a calidad de productos de software.
3. Cuenta con una amplia y detallada definición de los elementos de calidad de un

producto de software y además trae descrita una métrica en correspondencia para la determinación final de niveles de calidad.

De manera general se realiza una comparación de atributos de calidad que contempla cada modelo antes analizado en la tabla 1.

Tabla 1. Comparación de atributos propuestos por varios modelos de calidad [11]

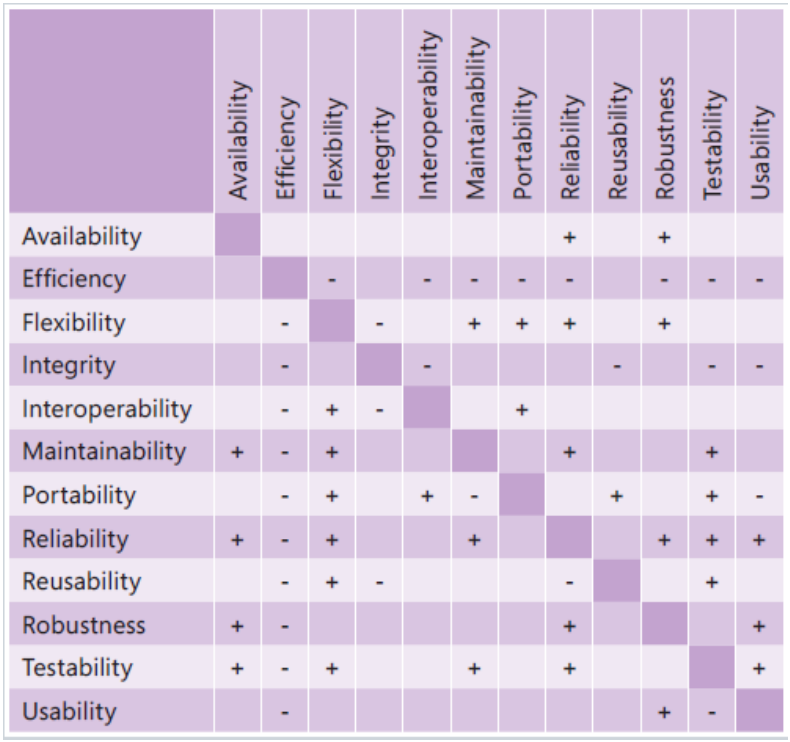
Característica de calidad	Boehm	McCall	ISO 9126
Capacidad de prueba	X	X	X
Corrección		X	
Eficiencia	X	X	X
Comprensibilidad	X		X
Fiabilidad	X	X	X
Flexibilidad		X	
Funcionalidad			X
Ingeniería humana	X		
Integridad		X	X
Interoperabilidad		X	X
Mantenibilidad	X	X	X
Posibilidad de cambio	X		
Portabilidad	X	X	X
Reusabilidad		X	

Si se realiza un análisis sobre la comparación plasmada en la tabla 1, es posible determinar que la ISO/IEC 9126 abarca la mayor cantidad de coincidencias en cuanto a cubrimiento de atributos de calidad. Teniendo en cuenta esto y los elementos expuestos en favor de esta norma, queda evidenciado el posible uso de este modelo para la evaluación arquitectónica de Cedrux.

1.1.2. Relaciones y conflictos entre atributos

La calidad del software, si bien se caracteriza por el logro de un conjunto de atributos, se considera alcanzada aún cuando todos los indicadores no se satisfacen en un ciento por ciento. Esto tiene sus bases en el hecho de que existe una estrecha relación entre todos los atributos, de manera que favorecer uno puede redundar en satisfacción o detrimento de otros, lo que se conoce como conflictos entre atributos.

En [17] se presenta el impacto mutuo que surge de la relación entre atributos presentes en un sistema mediante una matriz (figura 4). Téngase en cuenta que esta relación no se establece en función de los atributos definidos en la norma ISO/IEC 9126.



	Availability	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Reusability	Robustness	Testability	Usability
Availability								+		+		
Efficiency			-		-	-	-	-		-	-	-
Flexibility		-		-		+	+	+		+		
Integrity		-			-				-		-	-
Interoperability		-	+	-			+					
Maintainability	+	-	+					+			+	
Portability		-	+		+	-			+		+	-
Reliability	+	-	+			+				+	+	+
Reusability		-	+	-				-			+	
Robustness	+	-						+				+
Testability	+	-	+			+		+				+
Usability		-								+	-	

Figura 4. Matriz de relación entre atributos de calidad [17]

La lectura sobre la figura 4 se realiza desde dos perspectivas:

1. La fila

   Desde esta perspectiva se lee: *cuando se favorece el atributo A (fila) se afecta (positiva o negativamente) al atributo B (columna).*

2. Desde las columnas

   Desde esta perspectiva se lee: *el atributo A (columna) es afectado (positiva o negativamente) por el atributo B (fila).*

Es válido señalar que estas relaciones entre atributos hoy varían debido a que el avance de la tecnología tiende a ampliar el rango de posibilidades de interrelación. Un ejemplo de esto lo constituye el uso de marcos de trabajo que estén dirigidos al desarrollo de interfaces visuales para aplicaciones. Dichos marcos han sido creados sobre pautas de diseño y usabilidad de los recursos, sin embargo para su desarrollo se han usado lenguajes y tecnologías que propician a su vez una eficiencia en tiempo de respuesta. Esto evidentemente varía ligeramente, según sea el caso, lo planteado por la matriz de la figura 4 (usabilidad y

eficiencia pueden beneficiarse mutuamente con una decisión).

Lo que resulta invariable es el impacto de estos elementos en la evaluación del software. No tener en cuenta los conflictos que generan las decisiones arquitectónicas tomadas para el cumplimiento de un conjunto de atributos de calidad, puede acarrear ciclos infinitos de evaluación donde, cada vez que se pretenda mejorar el estado de un indicador se termine afectando otro irremediablemente.

Es fundamental en el desarrollo del software encontrar un balance de todos los atributos teniendo como punto de partida las relaciones naturales que se establecen entre estos. Un criterio para garantizar dicho cotejo, radica en identificar las prioridades de calidad en cada producto de software [17].

Acorde al Diccionario en línea de la Real Academia Española [19] el término “prioridad” significa:

1. Anterioridad de algo respecto de otra cosa, en tiempo o en orden.
2. Anterioridad o precedencia de algo respecto de otra cosa que depende o procede de ello.
3. Anterioridad o preferencia de algo respecto de otra cosa precisamente en cuanto es causa suya, aunque existan en un mismo instante de tiempo.

En términos de desarrollo de software y específicamente asociado a la calidad del software, las priorizaciones suelen hacerse de 3 formas distintas:

1. Priorización de requisitos no funcionales de atributos de calidad

A menudo los gerentes de proyectos e ingenieros del software se encuentran ante la disyuntiva de tener más requisitos que los que pueden realmente implementar, debido a restricciones que impone el desarrollo. Por lo tanto, es crucial distinguir las necesidades importantes de las menos importantes para garantizar su desarrollo [18].

Principales técnicas usadas: Asignación numérica (en grupos), Comparación por pares, Costo-valor, Votación acumulativa, Ranking [20].

2. Priorización de escenarios

Un escenario es una historia muy corta sobre una interacción con el sistema desde el punto de vista de las partes interesadas. Su priorización ayuda a identificar las cosas más importantes que se necesitan hacer para asegurar que el diseño esté en camino de alcanzar las principales preocupaciones de calidad y los valores solicitados por el cliente [21].

3. Priorización de atributos de calidad

Los atributos de calidad identificados son priorizados por su importancia. La importancia es definida, en base a las características del sistema o de los objetivos de la organización. La

prioridad del atributo afecta al orden de implementación de los requisitos de calidad. Los requisitos de calidad relacionados con los atributos más críticos son implementados antes [22].

1.2. Decisiones arquitectónicas en la construcción del software

La construcción de la arquitectura radica en la combinación de un conjunto de aserciones sobre los elementos que permiten constituir sus vistas en función de satisfacer los requisitos del software, tanto funcionales como no funcionales; dichas aserciones son comúnmente conocidas como decisiones y tienen mucho que ver con el logro de los niveles de abstracción y con la calidad del software.

Las decisiones arquitectónicas son presentadas por varios autores a través de conceptos diferentes.

Tácticas

Las tácticas han sido descubiertas por los investigadores del SEI a partir de su experiencia en el diseño de sistemas software. Se encuentran en un nivel de diseño más abstracto que el de los llamados estilos arquitectónicos. De hecho cada estilo implementa una o varias tácticas [49]. Se denomina colección de tácticas a una estrategia arquitectónica [23].

Bass, Clements y Kazman [23] abordan las tácticas como decisiones en función de favorecer atributos de calidad desde la perspectiva de lecciones aprendidas, o casos comunes. Ellos identifican dos enfoques del uso de tácticas:

- ✓ Las tácticas pueden refinar otras tácticas

  Este enfoque pretende ver las relaciones entre tácticas y los efectos de una táctica sobre varios atributos de calidad.
- ✓ Los patrones agrupan tácticas

  Esta variante concibe los patrones como un conjunto de tácticas para el logro de un beneficio de calidad.

Desde el punto de vista de escenarios arquitectónicos, las tácticas equivalen a elementos del *ambiente* que permiten controlar la *respuesta* al *estímulo*.

Estilos arquitectónicos

Los estilos y patrones de arquitectura son modelos estructurales definidos que presentan un esquema genérico demostrado con éxito para su solución [24].

Un estilo es un concepto descriptivo que define una forma de articulación u organización

arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales [25].

Al crear arquitecturas, los estilos estrechan el espacio de solución. En primer lugar, los estilos definen qué elementos pueden existir en una arquitectura (por ejemplo, componentes y conectores). En segundo lugar, los estilos definen las reglas para la forma de integrar estos elementos en la arquitectura. Por otra parte, los estilos abordan cuestiones no funcionales (por ejemplo, rendimiento), ya que muestran cualidades de calidad conocidas. Esto ayuda a estimar la calidad de un sistema antes de ser construido [26].

A diferencia de los patrones de diseño, que son centenares, los estilos se ordenan en seis o siete clases fundamentales y unos veinte ejemplares, como máximo [25].

Patrones de diseño

Un conjunto de autores ([27],[28] y [25]) reconoce como definición de patrón a la dada por Christopher Alexander [29]: *"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma."*

Por otra parte, una de las primeras definiciones, es la dada por Erich Gamma [30]: "*los patrones de diseño son descripciones de la comunicación de clases y objetos que son personalizadas para resolver un problema general de diseño en un contexto particular.*"

Esta conceptualización sitúa a los patrones de diseño en un nivel de abstracción más cercano a la implementación y por debajo de los estilos (tomando a los estilos como punto más alto de la pirámide conceptual de estructuración de un sistema). De aquí que los patrones de diseño constituyan un subconjunto a valorar en la toma de decisiones arquitectónicas que favorecerán la calidad de un sistema.

En opinión de la autora de este trabajo, la utilización de tácticas, estrategias, estilos, patrones, solo redunda en aplicación de conocimiento tácito a partir de buenas experiencias en el logro de calidad. La clave del éxito para cada caso estará fundamentalmente en la combinación de todos estos conceptos en base a la priorización y compensación de atributos de calidad que se tengan. Como elemento de aseguramiento, las decisiones arquitectónicas deben constar en el mecanismo de evaluación (dígase técnicas o métodos) de manera que una vez evaluados los resultados, pueda saberse con certeza qué decisión debe ser modificada.

1.3. Evaluación de arquitecturas de software

La concepción de la arquitectura de un software, como paso previo a la construcción del mismo, es un elemento clave en busca de eficiencia en el proceso de desarrollo. Desde 1968, Edsger Dikstra, gran exponente de la computación, propuso que se establezca una estructuración correcta de los sistemas de software antes de lanzarse a programar, escribiendo código de cualquier manera [25].

Siguiendo esta misma línea de pensamiento, a la arquitectura de un software se le atribuye la responsabilidad de lograr el cumplimiento de cualidades o características de calidad, más allá de lo meramente funcional (figura 5). Las decisiones arquitectónicas que se tomen estarán entonces dirigidas a la complacencia de los requisitos funcionales y no funcionales que establezca cada cliente.

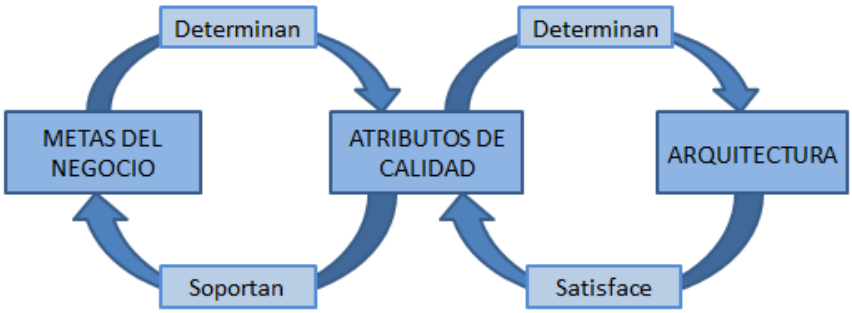


Figura 5. Relación entre elementos claves para la evaluación de la arquitectura [31]

Evaluar la arquitectura de un software es un medio para la obtención de indicadores de calidad del producto que bien pueden servir para:

- ✓ Determinar si el software que se está construyendo respalda los no funcionales que le fueron atribuidos.
- ✓ Decidir la mejor opción entre arquitecturas candidatas.
- ✓ Decidir cuál software, de varios posibles, respalda mejor los requisitos de un cliente.

La evaluación arquitectónica constituye el mecanismo de aseguramiento de calidad asociado al proceso de construcción de la arquitectura. Según se cita en [6] la evaluación sienta las bases para la toma de decisiones en situaciones tales como:

- ✓ Comparación de alternativas similares.
- ✓ Comparación de la arquitectura original y la modificada.
- ✓ Comparación de la arquitectura de software con respecto a los requisitos del sistema.
- ✓ Comparación de una arquitectura de software con una propuesta teórica.
- ✓ Valoración de una arquitectura en base a escalas específicas.

Si se toman en cuenta ambos enfoques entonces puede decirse que la evaluación arquitectónica tiene dos ventajas fundamentales:

1. *Genera una base de conocimientos para la toma de decisiones arquitectónicas.*
   Esta concepción es fundamental en el proceso de construcción de la arquitectura puesto que permite tomar y revisar las decisiones por iteraciones y determinar el cumplimiento de las metas esperadas del software, justo antes de comenzar a codificar.
2. *Genera un histórico de calidad asociado a productos.*
   Esto constituye el resumen de evaluaciones de productos, el cual puede ser usado como patrón de calidad a cumplir por software del mismo tipo.

Una vez definidos los objetivos de una evaluación, es fundamental escoger el mejor momento para realizarla. Una gran parte de los autores coinciden en que existen básicamente dos momentos en los cuales evaluar la arquitectura del software, así lo reafirma Hoffman [10] cuando expresa: *"La evaluación de una arquitectura puede ser ejecutada en diferentes etapas del proceso de creación de la misma. Actualmente se distinguen dos posibles fases: la evaluación temprana y la tardía."*

Este mismo autor establece que se evalúa tempranamente cuando sólo existen fragmentos de la descripción arquitectónica y por tanto no hay suficiente información tangible para determinar métricas o simular comportamientos [10].

La entrada fundamental para esta etapa de evaluación es la documentación de los requisitos, a partir de los cuales se pueden evaluar conceptualmente las decisiones arquitectónicas.

La evaluación tardía es ejecutada cuando existen elementos de diseño detallados que permiten evaluar en una amplia gama de posibilidades todos los requisitos y atributos de calidad[6] del sistema, incluso cuando algunos de los requisitos funcionales o no funcionales están implementados o pueden ser simulados.

Las condiciones de cada proyecto en que se construya software, determinarán la aparición de las fases de evaluación, pero sin dudas realizar ambas resultaría en la mejor de las prácticas de aseguramiento de la calidad en cuanto a construcción arquitectónica. Para que la evaluación sea satisfactoria en cada una de las fases es importante conocer las técnicas posibles y convenientes en cada caso.

[6] Se definen como las propiedades de un servicio que presta el sistema a sus usuarios, constituyen también criterios de medida de la calidad del software (14). También se les conoce como factores u objetivos de calidad.

1.3.1. Técnicas de evaluación

Jan Bosh, exponente de la corriente de Arquitectura basada en escenarios, plantea que el aseguramiento de la calidad arquitectónica se mueve en dos dimensiones fundamentales [32]:

1. *Orientado a la arquitectura*: Donde se evalúa la arquitectura por las partes interesadas y por criterios de expertos.
2. *Enfocado a los atributos de calidad*: Donde bien se evalúa cualitativamente, en función de comparaciones, o bien se evalúa cuantitativamente, en función de predicciones.

En la opinión del autor de este trabajo existe una trazabilidad que une estas dos dimensiones en una. En una reflexión histórico-lógica, se puede afirmar que las partes interesadas evaluarán la arquitectura en función del cumplimiento visible o medible de sus requisitos no funcionales (RNF) y los expertos evaluarán la interacción de las decisiones arquitectónicas con respecto a combinaciones anteriores conocidas. Ambos análisis de los evaluadores (interesados y expertos) tienen su denominador común en los atributos de calidad debido a que:

- ✓ Un requisito no funcional encuentra su clasificación en la definición de un atributo de calidad.
- ✓ Las decisiones arquitectónicas siempre pueden ser agrupadas y tomadas en función de los atributos de calidad que favorecen o afectan.

Es por esto que cuando se habla de cómo evaluar siempre se hace referencia a las técnicas de evaluación como una categoría que determina la manera de medir el estado de una arquitectura. Las clasificaciones de las técnicas de evaluación de arquitectura se muestran en la figura 6.

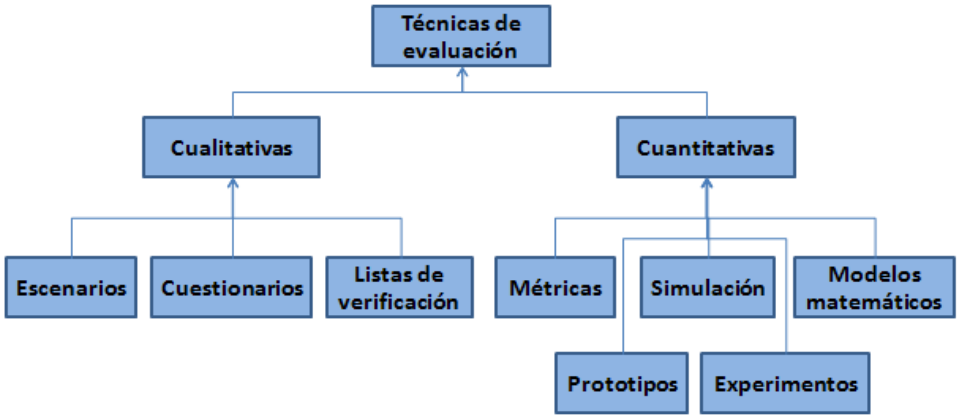


Figura 6. Clasificación de las técnicas de evaluación [8]

Las técnicas cuantitativas tienden a tener más aceptación en cuanto a la obtención de valores,

sin embargo tienen en su contra la necesidad de herramientas que las soporten, porque en tiempo de producción, se hace muy engorroso ponerlas en práctica de manera manual. Otro elemento que determina su uso es el nivel de especificación y construcción de la arquitectura, por lo cual esta técnica es utilizada en la fase tardía de evaluación.

Las técnicas cualitativas, en cambio, permiten determinar estados de las arquitecturas en términos no contables, o sea, en valoraciones de las decisiones arquitectónicas o de los escenarios arquitectónicos e incluso de los atributos de calidad. Muchas veces estos métodos resultan los más usados por su aplicabilidad y su posibilidad de evaluar la mayor parte de los indicadores de calidad de un producto de software.

En consideración del autor del trabajo, la aplicación de cada técnica está sujeta a las condiciones del proceso de construcción de la arquitectura que se desarrolle y de los atributos de calidad a evaluar en cada producto. Sin embargo, la mejor de las condiciones sería la particularización y combinación de las técnicas idóneas para evaluar cada atributo de calidad.

1.3.2. Métodos de evaluación de arquitectura

A la par de las técnicas de evaluación se han descrito numerosos métodos, cuya finalidad mayor radica en identificar fases, pasos y roles para la realización de la evaluación arquitectónica. Según remarca Losavio en [33]: "*Un estudio comparativo de tales métodos favorece la identificación de los procedimientos y actividades que mejor respondan al complejo proceso de generar una arquitectura en función de un conjunto de requisitos iniciales*".

Realizando una revisión de algunos de los métodos que hoy existen (tabla 2), se pueden identificar tres escuelas representativas en la identificación y descripción de dichos métodos: el Instituto de Ingeniería de Software (SEI[7], del inglés) de la Universidad Carnegie Mellon, las universidades holandesas (fundamentalmente con sedes en Amsterdam y Eindhoven), y la Universidad Simón Bolívar de Venezuela.

Tomando en cuenta que:

1. ATAM constituye el método más referenciado en la bibliografía consultada en esta investigación, además de ser el método de evaluación por excelencia en el modelo de desarrollo arquitectónico del SEI.
2. MECABIC adapta diferentes elementos de algunos métodos de evaluación como ATAM, ARID, Losavio y otros [34] y adiciona el enfoque de Arquitecturas de Software Basadas en Componentes (ASBC).

[7] Software Engineering Institute

3. QUASAR es el método más actual de los referenciados en la tabla 2, está respaldado por el SEI y permite emplear variadas técnicas de evaluación.

Se analizan entonces, con más detalle, estos tres métodos como muestra representativa de los métodos existentes.

Tabla 2. Resumen de métodos de evaluación de arquitectura de software

Método	Referencia	Objetivo	País
SAAM	[35]	Determinar facilidad de modificación.	USA
ATAM	[36]	Determinar riesgos potenciales, puntos de sensibilidad y los puntos de desventaja.	USA
ARID	[37]	Realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo.	USA
CBAM	[38]	Determinar costo y/o beneficio de las decisiones arquitectónicas	USA
FAAM	[39]	Determinar interoperabilidad y extensibilidad de familias de sistemas de información.	Holanda
ALMA	[40]	Determinar facilidad de modificación desde 3 posibles metas.	Holanda
PASA	[41]	Determinar rendimiento.	USA & México
SALUTA	[42]	Determinación del nivel de usabilidad (facilidad de uso) que provee una arquitectura a un sistema.	Holanda
MECABIC	[34]	Evaluar y analizar la calidad exigida por los usuarios de Arquitecturas de Software Basadas en Componentes (ASBC).	Venezuela
QUASAR	[43]	Determinar cumplimiento de requisitos no funcionales de atributos de calidad.	USA

Método de Análisis de Compensación de la Arquitectura (ATAM[8], del inglés)

El propósito fundamental de ATAM es evaluar las consecuencias de las decisiones de arquitectura en función de los requisitos de atributos de calidad. ATAM está llamado a ser un método de identificación de riesgos, un medio para detectar áreas de riesgo potencial dentro de la arquitectura de un sistema complejo de software [36].

Sin embargo, lo que distingue a ATAM del resto de los métodos de evaluación basados en escenarios, es su enfoque hacia la compensación de decisiones arquitectónicas que afectan a más de un atributo de calidad en la construcción de la arquitectura. Este punto de vista lo cumple a partir de la utilización del denominado árbol de utilidad, que es el mecanismo a través del cual se elicitan los atributos de calidad que determinan la utilidad del sistema, se desglosan hasta el nivel de escenarios y se priorizan [6].

ATAM, para la evaluación cualitativa que propone, establece un conjunto de fases y pasos que son mostrados en síntesis en la figura 7 y descritos en el Anexo 1.

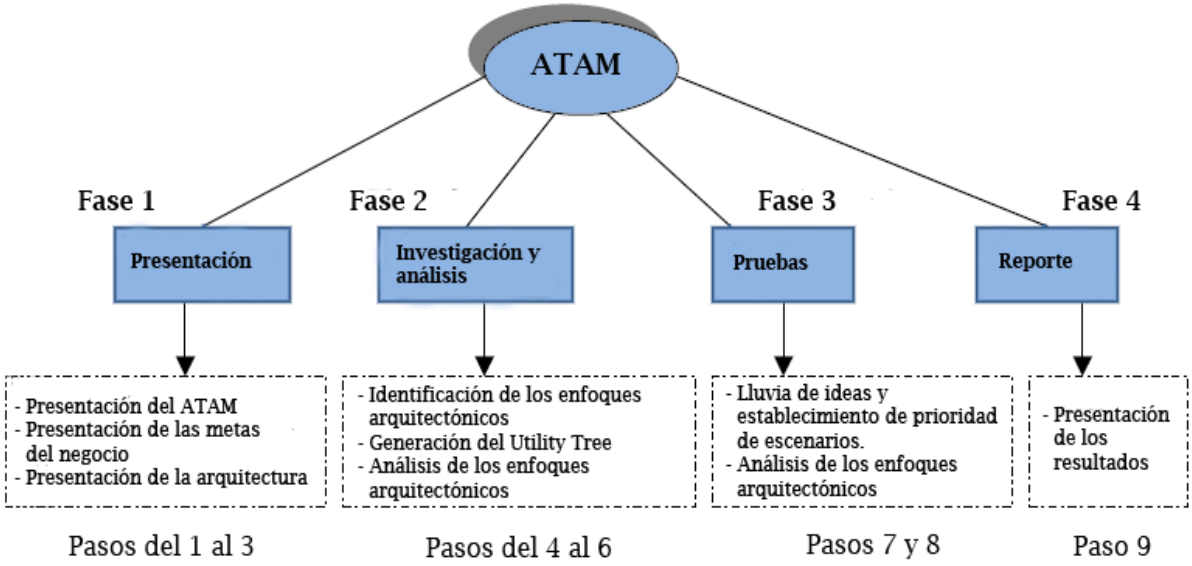


Figura 7. Principales fases y pasos de ATAM [44]

El autor de este trabajo analiza algunas deficiencias en la estructuración de ATAM:

1. *No existe una trazabilidad directa de requisito no funcional (o requisito de atributos de calidad) a escenarios arquitectónicos.*

   En la especificación del método ATAM se habla de la presencia de los requisitos de atributos de calidad y se conceptualizan los escenarios, sin embargo no se refiere a la

[8] Architecture Tradeoffs Analysis Method

transformación directa de un requisito no funcional en términos de un escenario.

2. *No existen nociones de entradas del proceso de construcción a la aplicación del método.*

   La descripción del método no refiere qué artefactos de entrada se requieren para su total funcionamiento. En las actividades iniciales del proceso se refiere solo a la presentación por un arquitecto de la arquitectura, no establece pautas específicas de qué elementos de la arquitectura se necesitan ni en qué formato.

Método de Evaluación de Arquitecturas de Software Basadas en Componentes (MECABIC)

MECABIC está inspirado en ATAM, aunque con el objetivo de facilitar su aplicación sobre ASBC se incluyó en algunos de sus pasos un enfoque de integración de elementos de diseño, inspirado en la construcción de partes arquitectónicas adoptado por el Diseño Basado en Arquitectura (ABD[9], del inglés). Además presenta un árbol de utilidad inicial basado en el modelo de calidad ISO-9126 instanciado para arquitectura de software propuesto por Losavio [34].

Este método propone un conjunto de escenarios tipo con enfoque a ASBC, como se muestra en la tabla 3.

Tabla 3. Subconjunto del árbol de utilidad inicial propuesto por el método [34]

Característica	Sub-característica	Escenario
Funcionalidad	Interoperabilidad	El sistema posee componentes capaces de leer datos provenientes de otros sistemas.
		El sistema posee componentes capaces de producir datos para otro sistema.
	Precisión	Los resultados ofrecidos por los componentes del sistema son exactos.
		La comunicación entre los componentes no altera la exactitud de los datos
	Seguridad	El sistema detecta la actuación de un intruso e impide acceso a los componentes que manejen información sensible
		El sistema asegura que los componentes no pierdan datos ante un ataque (interno o externo).
	Obediencia	Los componentes respetan un estándar de fiabilidad.
		La comunicación entre los componentes no viola los estándares de fiabilidad.
Fiabilidad	Madurez	Los componentes del sistema manejan entradas de datos incorrectas.
	Tolerancia a fallas	Todas las operaciones ejecutadas por los

[9]Architecture Based Design

		componentes se realizan correctamente bajo condiciones adversas.
	Capacidad de restablecimiento o recuperación	Los componentes del sistema no fallan bajo ciertas condiciones especificadas.
		Ante problemas con el ambiente un subconjunto determinado de los componentes puede continuar prestando sus servicios.
Eficiencia	Tiempo de comportamiento	El sistema debe recibir los servicios de sus componentes en el transcurso de un tiempo indicado.
	Recursos utilizados	Los componentes pueden compartir recursos adecuadamente.
		El sistema controla que ningún componente se quede sin recursos cuando los necesita.
Mantenibilidad	Habilidad de cambio, estabilidad, prueba	Es posible verificar el estado de los componentes del sistema.
		El sistema brinda facilidad para adaptar un componente.
		El sistema debe facilitar la sustitución/adaptación de un componente.
Portabilidad	Adaptabilidad	El sistema debe continuar funcionando correctamente aun cuando los servicios de los componentes provistos por el ambiente varíen.
	Capacidad de Instalación	Los componentes pueden instalarse fácilmente en todos los ambientes donde debe funcionar.
	Co-existencia	Los componentes manejan adecuadamente recursos compartidos del sistema.

A consideración de la autora de este trabajo, el método MECABIC constituye un conjunto de buenas prácticas y realiza un aporte en la construcción de los escenarios del sistema a partir del punto de vista de ASBC. Como elemento fundamental a su favor está la elección del modelo de calidad que rige la evaluación, elemento que la mayor parte de los métodos no incluye.

Método para la Valoración de Calidad de Arquitecturas de un Sistema Software (QUASAR)

QUASAR es considerado, más que un método de evaluación de arquitecturas de software, como un método de aseguramiento de la calidad arquitectónica. No puede decirse que se acoja a una técnica de evaluación específica, sino que por el contrario asimila cualquiera que sea más conveniente para cada caso.

La esencia de QUASAR radica en la creación de un expediente de calidad arquitectónica elaborado por los arquitectos de sistema del proyecto para ser presentado a los equipos de evaluación. La estructura de este expediente se compone de casos de calidad, los cuales son

una colección coherente de alegaciones, argumentos y pruebas (figura 8) que hace que los desarrolladores afirmen que sus productos tienen la calidad suficiente [37].

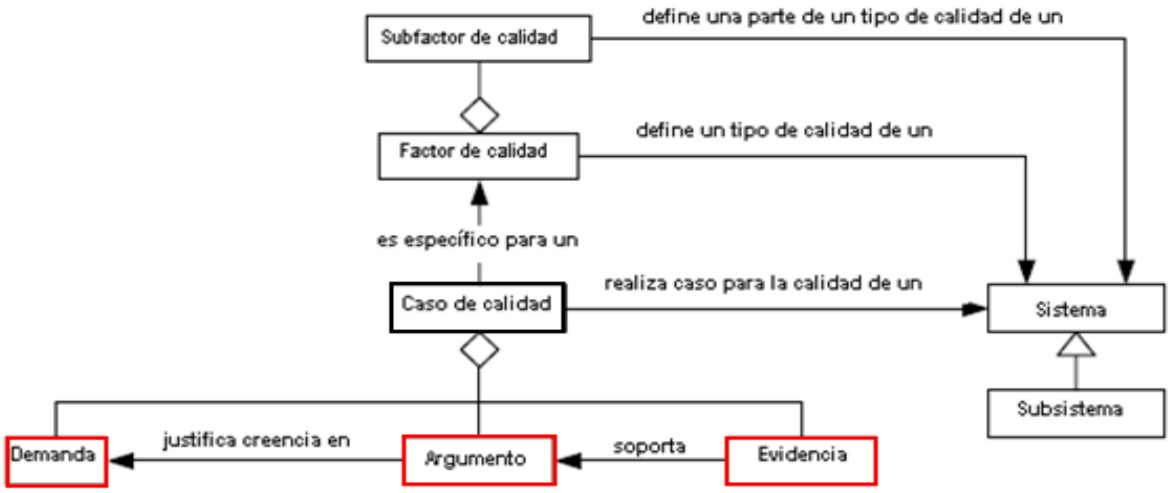


Figura 8. Estructura de un caso de calidad [46]

Como se evidencia en rojo en la figura 8, los casos de calidad se constituyen a partir de tres elementos fundamentales [43]:

1. Las *demandas* de los arquitectos de que sus productos tienen suficiente calidad, teniendo en cuenta que la calidad se define en términos de los requisitos y atributos de calidad definidos en el modelo oficial de calidad del proyecto. Las demandas están relacionadas a uno o varios atributos de calidad y a un grupo de uno o más requisitos de calidad asociados. Estos se escriben atendiendo a atributos de calidad (por ejemplo, rendimiento) y sus subatributos (si existen). Ver representación gráfica en Anexo 2.
2. Los *argumentos* son una combinación de:
   - ✓ Decisiones arquitectónicas que involucran:
     1. El uso de patrones, estilos y mecanismos arquitectónicos.
     2. La incorporación de un componente específico.
     3. El uso de una forma específica en la que los componentes arquitectónicos deben colaborar para alcanzar los atributos de calidad asignados y los requisitos de calidad.
   - ✓ Razones de por qué las decisiones arquitectónicas son adecuadas y suficientes.

   Ver representación gráfica en Anexo 3.

3. La *evidencia* soporta los argumentos y demuestra que:
    - ✓ Se tomaron las decisiones arquitectónicas que se argumentaron.
    - ✓ Las decisiones tomadas fueron apropiadas y suficientes para respaldar la creencia en las demandas.

    Ver representación gráfica en Anexo 4.

Se puede decir que QUASAR cuenta estas entre sus ventajas:

1. Utiliza el enfoque jerárquico de grandes sistemas de software como parte de las consideraciones para determinar los casos de calidad a evaluar en cada iteración.
2. Permite evaluación tardía o temprana, en cualquiera de los casos lo que variará serán los argumentos y evidencias, según nivel de completitud de la arquitectura.
3. Promueve la confección de un artefacto donde se consolide la información arquitectónicamente significativa en términos de evaluación, de manera que la realización de esta por el equipo validador sea mucho más fácil.

Sin embargo, QUASAR presenta deficiencias que atentan contra su mejor uso y que aún hoy constituyen las direcciones futuras en su desarrollo, entre ellas se cuentan [47]:

1. No contempla la garantía de la calidad arquitectónica entre múltiples subsistemas: La elaboración de los casos de calidad se realiza por subsistemas independientes, de manera que las metas globales de calidad y la contribución de cada subsistema a las mismas nunca se contempla.
2. No considera los conflictos de atributos de calidad: esto implica una brecha en el análisis de las decisiones arquitectónicas que tengan implicaciones en más de un atributo o subatributo de calidad.
3. No contempla mecanismos para la determinación de “suficiente” calidad arquitectónica: QUASAR plantea cómo ordenar el proceso de evaluación pero no cómo retroalimentarlo a partir de las evaluaciones, de manera que no basta con obtener un valor de indicador, ya sea cualitativo o cuantitativo, sino tener la magnitud de variación o aserción de las decisiones arquitectónicas tomadas.

Tras el análisis de cada uno de los métodos se puede realizar un estado comparativo teniendo en cuenta un conjunto de indicadores, como se muestra en la tabla 4.

Tabla 4. Comparación de los métodos analizados

	ATAM	**MECABIC**	**QUASAR**
Objetivo	Evaluar las consecuencias de las decisiones de	Evaluar sobre la base de ATAM, incluyendo el	Presentación de casos de calidad a equipos de evaluación, para

	arquitectura en función de los requisitos de atributos de calidad	enfoque de ASBC y el modelo de calidad ISO-9126.	determinar cumplimiento de la arquitectura con sus atributos de calidad y requisitos asociados.
Atributos de calidad contemplados.	Todos los que se definan para el producto.	Todos los que se definan para el producto.	Todos los que se definan para el producto.
Objetos analizados	1. Decisiones arquitectónicas 2. Atributos de calidad 3. Escenarios arquitectónicos	1. Decisiones arquitectónicas 2. Atributos de calidad 3. Escenarios arquitectónicos (con enfoque de ASBC)	1. Demandas 2. Argumentos 3. Evidencias (Equivale a Atributos de calidad- RNF-decisiones)
Técnica de evaluación	Escenarios	Escenarios	Cualesquiera que se escojan.
Etapa del proceso	Luego de que el diseño de la arquitectura ha sido establecido.	Luego de que el diseño de la arquitectura ha sido establecido.	Luego de que el diseño de la arquitectura ha sido establecido.
Enfoque	1. Árbol de utilidad 2. Puntos sensibles 3. Puntos de balance 4. Riesgos	1. Árbol de utilidad (ASBC) 2. Puntos sensibles 3. Puntos de balance 4. Riesgos	1. Casos de calidad. 2. Jerarquía de sistema.

Tras el análisis de los métodos y la comparación resumen de la tabla 4 es posible plantear que el método QUASAR constituye la mejor variante para aplicar en la investigación actual, ya que posee factores claves como la posibilidad de emplear varias técnicas para la evaluación de los indicadores de calidad, permitiendo gradualmente la asimilación de mejoras y perfeccionamiento del proceso de evaluación sin generar cambios drásticos visibles en la metodología de trabajo. Además, este método define grupos y formas de trabajo, no ajenas ni complicadas de implantar en las condiciones actuales del proyecto ERP-Cuba.

Sin embargo, para su aplicación efectiva se hace necesario realizarle mejoras que vayan dirigidas a solucionar las deficiencias antes descritas y que permitan también especificar su papel dentro del proceso de construcción de la arquitectura de un sistema ERP.

En lograr dichos avances de este método, para su aplicación en el proyecto ERP-Cuba durante la evaluación de Cedrux, se centra este trabajo.

1.4. Conclusiones parciales

Tras el análisis de los elementos conceptuales que caracterizan a la evaluación de arquitectura de software y a los sistemas ERP se puede arribar a las siguientes conclusiones:

1. Los ERP son sistemas grandes y voluminosos, modulares e integrados, lo cual

conlleva a que construir y evaluar sus arquitecturas sea complejo.

2. Las decisiones arquitectónicas son las mejores armas de un arquitecto para lograr la calidad en la construcción de la arquitectura, fundamentarlas sobre la base de buenas prácticas es clave para alcanzar el éxito.
3. Evaluar la calidad de la arquitectura durante todas sus fases es una práctica recomendable en función de lograr la mejor línea base del software posible, sin embargo no debe perderse de vista que la evaluación siempre comprenda la verificación de lo solicitado por el cliente.
4. El empleo de las técnicas cuantitativas y cualitativas deberá estar sujeta a las condiciones del proceso de construcción de la arquitectura que se desarrolle y de los atributos de calidad a evaluar en cada producto.
5. El método QUASAR constituye la mejor variante para aplicar en la investigación actual, ya que permite emplear varias técnicas para la evaluación de los indicadores de calidad, permitiendo gradualmente la asimilación de mejoras y perfeccionamiento del proceso de evaluación sin generar cambios drásticos en la metodología de trabajo.
6. Para su aplicación efectiva se hace necesario realizarle mejoras al método QUASAR que permitan medir la calidad integral de la arquitectura y tener en cuenta el impacto de decisiones arquitectónicas en múltiples indicadores.

CAPÍTULO 2. PROPUESTA DE VARIACIÓN A QUASAR

El presente capítulo aborda las variaciones realizadas a QUASAR, fundamentalmente referidas a los mecanismos propuestos para la verificación de la calidad integral de sistemas modulares, y a los conflictos que generan las relaciones entre atributos asociadas a las decisiones arquitectónicas tomadas para favorecerlos. Finalmente se enuncia el impacto de lo propuesto sobre las predefiniciones del método.

2.1. Consideraciones generales de la propuesta

El desarrollo de la evaluación de la arquitectura es un proceso compuesto de una secuencia de actividades, cuyo objetivo final es conocer el nivel de aserción de las decisiones arquitectónicas tomadas para el logro de los requisitos asociados al sistema.

Para este trabajo, la organización del conjunto de actividades, roles y los artefactos que intervienen en el proceso de evaluación estarán determinados por las definiciones de QUASAR, teniendo en cuenta las particularidades de este método en cuanto a que:

1. Utiliza el enfoque jerárquico de grandes sistemas de software como parte de las consideraciones para determinar los casos de calidad a evaluar en cada iteración.
2. Permite evaluación tardía o temprana, en cualquiera de los casos lo que variará serán los argumentos y evidencias, según nivel de completitud de la arquitectura.
3. Promueve la confección de un artefacto donde se consolide la información arquitectónicamente significativa en términos de evaluación, de manera que la realización de esta por el equipo validador sea mucho más fácil.
4. Se adecúa a la estructura organizativa vigente en el proyecto ERP-Cuba y a los modelos de desarrollo de base para la construcción de Cedrux.

El proceso de evaluación se considera para esta propuesta como iterativo e incremental, acorde al proceso de construcción de la arquitectura en el que está inmerso. Es por esto que se propone la construcción de un expediente de evaluación en pasos sucesivos en función del avance de la fase arquitectónica, según lo planteado por QUASAR.

Teniendo en cuenta la propuesta metodológica para la obtención de los componentes de Cedrux contenida en [48], el proceso de evaluación que plasma esta investigación afectaría directamente las siguientes actividades de dicha propuesta:

1. Actividad #1: Análisis de los artefactos generados en las fases de modelado de negocio y análisis del sistema.

   Esta es la actividad inicial del flujo para la obtención de los componentes de Cedrux, de aquí que constituya una entrada para la misma el denominado *Árbol de utilidades de*

atributos de calidad, cuya confección se propone en esta investigación (véase el epígrafe 2.3.1).

2. Actividad #5: Análisis de los atributos de calidad y escenarios arquitectónicos.
   Esta actividad persigue el análisis de los atributos de calidad a favorecer con la solución, siendo el punto de partida para ello los requisitos del software y las restricciones de diseño.
   Para esta actividad se realizan algunas acotaciones con la actual propuesta, puesto que se presenta el análisis de las restricciones de diseño a partir de RNF de atributos de calidad. Además, se propone un algoritmo de base para el análisis de atributos de calidad y sus relaciones, en función de orientar la toma de decisiones arquitectónicas que conforman la solución (véanse los epígrafes 2.2 y 2.3.2).

Teniendo en cuenta el análisis realizado en 1.3.2 sobre los elementos aún no presentes en QUASAR se exponen entonces los elementos que suplirán estas carencias.

2.2. Calidad integral de la arquitectura

Si se considera la representación del sistema como un árbol, a partir de su descomposición en niveles de abstracción (figura 9), los criterios que se proponen para las iteraciones de evaluación son:

1. *Recorridos horizontales por subsistemas*.
2. *Recorridos verticales por subsistemas*.

Estos criterios favorecen la evaluación en cualquier fase de construcción, haciendo uso de la descomposición de los sistemas ERP en subsistemas y/o componentes (niveles de empaquetamiento).

Sin embargo, como muestra la figura 9, la calidad integral del sistema es difícil de determinar una vez que se integren subsistemas en las fases de evaluación. Aunque un sistema posee un conjunto de RNF de atributos de calidad de manera general (que afectan a todo el sistema), también posee un conjunto de RNF a nivel de subsistemas, que hacen que la calidad integral del sistema varíe en la medida en que este vaya integrando sus partes.

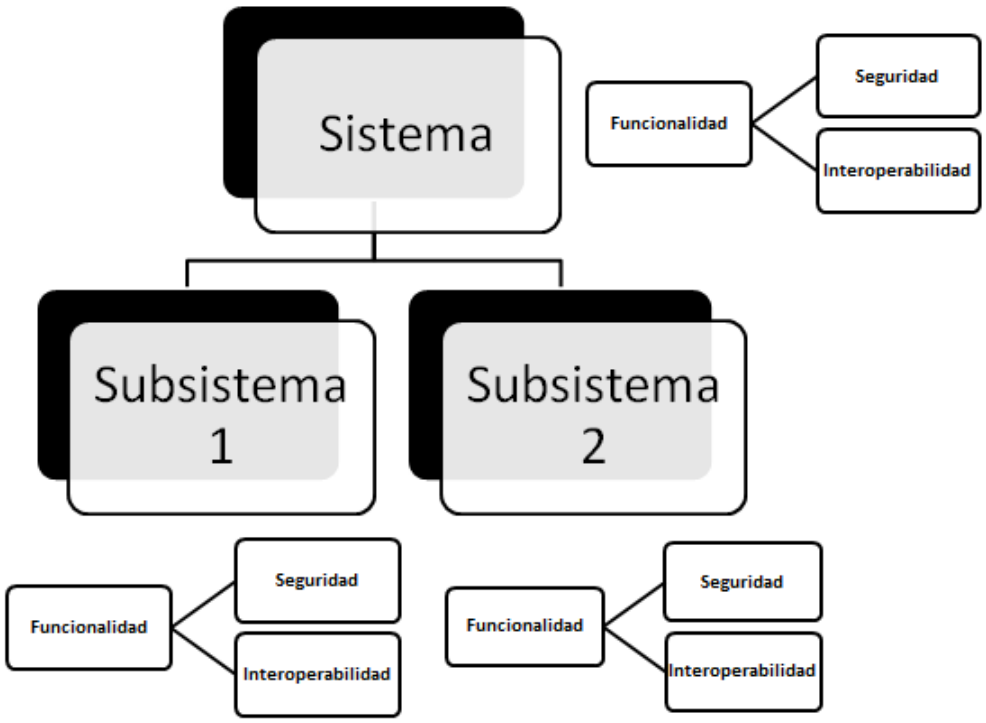


Figura 9. Comportamiento de la calidad de un sistema y su descomposición jerárquica en subsistemas

En función de corregir esto, se propone realizar evaluaciones de subsistemas independientes en cada iteración y concebir, en una fase de pruebas de integración, la creación de un caso de calidad de sistema que tenga en cuenta:

1. Requisitos de atributos de calidad que hayan sido identificados de manera global para el sistema.
2. Las dimensiones no funcionales (asociadas a atributos de calidad) de *requisitos funcionales de integración*[10].

Cuando se intenta ordenar el proceso de evaluación de la arquitectura, se hace imprescindible tener una claridad y precisión en las medidas de calidad que el software debe tener. Estos indicadores se desglosan, priorizan y miden a partir del conocimiento de los RNF del software. Para poder cumplir los tres aspectos a tener en cuenta para la creación del caso de calidad de sistema se hace necesario entonces establecer un patrón para la construcción de los requisitos de atributos de calidad, el cual se describe en el epígrafe 2.2.1.

2.2.1. Requisitos no funcionales de atributos de calidad

El principal objetivo de la evaluación propuesta por QUASAR es determinar la calidad de la

[10]Dícese de aquellos requisitos que implican funcionalidad o datos de más de un subsistema o componente.

arquitectura en términos del grado en que esta ayuda al sistema a alcanzar sus metas de calidad y requisitos asociados [47]. De aquí se infiere la importancia que el método atribuye a la necesidad de identificación y descripción de los requisitos no funcionales, como base para la evaluación arquitectónica de un sistema de software.

Con el objetivo de estandarizar el proceso de evaluación, a partir de la determinación de los indicadores medibles de la calidad, se propone entonces establecer un patrón para la redacción de los RNF de atributos de calidad, como sigue:

[*Criterio de sistema*] [*aspecto a lograr*] [*medida de respuesta*]

Donde:

1. El *criterio de sistema* puede ser:
   - ✓ Interfaz
   - ✓ Funcionalidad de sistema (RF)
   - ✓ Componente
   - ✓ Aplicación
2. El *aspecto a lograr* se refiere al estado que se pretende tenga el *criterio de sistema* y que a su vez se interpreta como “criterio de calidad” para el atributo al cual esté asociado el requisito.
3. La *medida de respuesta* es el valor cuantificable del *aspecto a lograr* haciendo medible el requisito.

La definición de este patrón permite garantizar que se escriban correctamente los RNF de atributos de calidad, de manera que la evaluación se sustente sobre elementos medibles.

Ejemplos:

1. El sistema manejará mecanismos de encriptación para las contraseñas de los usuarios (Seguridad).
   - ✓ Criterio de sistema: el sistema
   - ✓ Aspecto a lograr: contraseñas
   - ✓ Medida de respuesta: encriptadas
2. El sistema debe soportar hasta 40 conexiones simultáneas (Eficiencia).
   - ✓ Criterio de sistema: el sistema
   - ✓ Aspecto a lograr: conexiones simultáneas
   - ✓ Medida de respuesta: 40
3. El idioma de todas las interfaces de la aplicación será el español (Usabilidad).
   - ✓ Criterio de sistema: las interfaces
   - ✓ Aspecto a lograr: idioma

- ✓ Medida de respuesta: español

Para no dejar fuera ninguna dimensión de interés para el logro de la calidad, se propone realizar un análisis de requisitos no funcionales de otras clasificaciones, como son: requisitos legales, requisitos de hardware y software (tecnológicos), requisitos económicos, entre otros. Tras la revisión de cada uno de estos requisitos, se deben identificar restricciones que los mismos impongan al diseño o estructuración de la aplicación a desarrollar, y estas restricciones deben incluirse como RNF de atributos de calidad, en sus respectivas agrupaciones. Dichos requisitos deben cumplir también con el patrón antes establecido.
Para la generación del caso de calidad de sistema propuesto en 2.2, se persigue identificar los RNF de atributos de calidad globales del sistema, el listado de estos requisitos se alimenta de dos fuentes principales:

1. Los requisitos que han sido establecidos para el producto.
2. Los requisitos que son comunes a todos los subsistemas que integran el producto.

Siguiendo esta forma estándar para la descripción de los RNF de atributos de calidad es posible tratar los requisitos de los productos de software que cumplan con las características antes descritas; para ello sería necesario considerar:

- ✓ Los RNF de atributos de calidad de los subsistemas integrados en un producto que coincidan completamente (las 3 partes del patrón) en cada subsistema, pasan a ser requisitos generales del producto.
- ✓ Cuando en los subsistemas que componen al producto existen RNF de atributos de calidad, clasificados en un mismo subatributo, y varía alguna de las partes, se consideran requisitos diferentes.
- ✓ Los requisitos cuyas dos primeras partes (según el patrón) se mantengan iguales en todos los subsistemas del producto y solo varíe la medida de respuesta, deben incluirse en el producto pero con una nueva medida a establecer.

2.3. Los conflictos de atributos y los casos de calidad

Uno de los retos de la evaluación de arquitectura radica en la identificación de elementos que “desequilibran” la arquitectura, de ahí que la retroalimentación que produce la evaluación hacia la construcción, sea orientada al logro del “balance” de calidad.

2.3.1. Priorización

El primer paso en la búsqueda de un equilibrio de los atributos de calidad potenciados con las decisiones arquitectónicas, lo constituye la priorización de dichos atributos de manera que:

1. Posibilite la toma de decisiones arquitectónicas, para el logro del equilibrio de los factores de calidad, según su orden de prioridad.
2. Establezca un posible orden de prueba del cumplimiento de los atributos de calidad durante la evaluación, en función de prioridades.

Teniendo en cuenta estos elementos, se propone establecer un mecanismo cuantitativo para la priorización que combine los criterios de todos los interesados[11].

Priorización de atributos por subsistemas

La priorización se realizará a partir de los RNF de atributos de calidad, los cuales se listarán y someterán a votación por cada uno de los interesados y arquitectos. Esta actividad deberá ser llevada a cabo en un paso previo a la evaluación, incluso se recomienda sea efectuado a inicios del desarrollo arquitectónico. La votación seguirá un criterio básico: *Importancia de lograr el requisito*.

Para los efectos de esta investigación, la *Importancia de lograr el requisito,* es un concepto que va a estar sujeto a la posición de cada votante con respecto al sistema, la cual genera intereses diversos. Es por esta razón que, los criterios subjetivos de cada persona involucrada en la priorización no son manejados sino el interés final de los mismos en el logro de cada requisito, en una escala sujeta a su nivel de importancia.

El rango de valores admisibles para este criterio es:

5: Muy importante

4: Importante

3: Medianamente importante

2: Poco importante

1: No importante

Del desglose anterior se deduce que: a mayor valor, mayor importancia.

Una vez obtenidos los votos de cada uno de los interesados, se tendría una priorización de lo que se pudiera denominar “hojas” en el contexto de un árbol de calidad tal y como lo representa la ISO/IEC 9126-1, dicho árbol estaría compuesto (en orden descendente) por: atributos, subatributos y RNF de atributos de calidad con sus respectivas priorizaciones. Esto

[11] Los interesados en el proyecto son personas y organizaciones que participan de forma activa en el proyecto o cuyos intereses pueden verse afectados positiva o negativamente como resultado de la ejecución del proyecto o de su conclusión. Se incluyen los clientes, patrocinadores, la organización ejecutante y el público [24].

quedaría:

1. Funcionalidad
    1.1. Seguridad
        1.1.1. _5_ El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.
        1.1.2. _5_ El sistema manejará mecanismos de encriptación para las contraseñas de los usuarios.
    1.2. Interoperabilidad
        1.2.1. _3_ El sistema almacenará información proveniente del Sistema Mantenimiento Vehicular, al mismo tiempo que enviará información a este, todo de manera automática y transparente para el usuario final.
        1.2.2. _4_ El sistema permitirá importar información en los formatos Word y Excel.

Luego, se hace necesario conocer las prioridades, que se irán derivando de las atribuidas mediante votación, para el resto de los niveles del árbol. Para ello, en el caso del cálculo de prioridad de los subatributos se emplea el criterio de la *media aritmética*, expresado mediante la fórmula:

$\bar{X} = (\sum X_i * f_i) / f_i$

Siendo:

X_i: Valores de importancia

f_i: Cantidad de requisitos asociados a cada valor

Siguiendo el ejemplo del árbol anterior, el cálculo quedaría como se muestra en la tabla 5.

Tabla 5. Uso de la media aritmética para el cálculo de prioridad del subatributo Seguridad

Rango Priorización x_i	**Cantidad requisitos** f_i	**Valores Resultantes** $X_i * f_i$
1 – 1,99	0	0
2 – 2,99	0	0
3 – 3,9	0	0
4 – 4,9	0	0
5	2	10
	2	10

Tomando como *Rango de Priorización* los posibles valores que se encuentran en la priorización de todos los requisitos.

Para el ejemplo mostrado en la tabla 6 el cálculo de la prioridad del subatributo Seguridad quedaría:

$\bar{X} = (\sum X_i * f_i) / f_i$

$\bar{X} = 10,0 / 2$

$\bar{X} = 5,00$

Análogamente se calcula la media para el subatributo Interoperabilidad.

Completando el árbol se tendría:

1. Funcionalidad
    1.1. 5,00 Seguridad
        1.1.1. 5 El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.
        1.1.2. 5 El sistema manejará mecanismos de encriptación para las contraseñas de los usuarios.
    1.2. 3,50 Interoperabilidad
        1.2.1. 3 El sistema permite importar información proveniente de sistemas externos para operar con ella, al mismo tiempo que permite exportar datos a sistemas externos.
        1.2.2. 4 El sistema permitirá importar información en los formatos Word y Excel.

Para el cálculo de la prioridad de los atributos, como máximo nivel del árbol, se hace uso de la media aritmética, esta vez teniendo en cuenta las prioridades de todos los requisitos asociados a sus subatributos. Siguiendo la ejemplificación del árbol anterior, quedaría como en la tabla 6.

Tabla 6. Uso de la media aritmética para el cálculo de prioridad del atributo Funcionalidad

Rango Priorización x_i	**Cantidad requisitos** f_i	**Valores Resultantes** $X_i * f_i$
1 – 1,99	0	0
2 – 2,99	0	0
3 – 3,99	1	3

4 – 4,99	1	4
5	2	10
	4	17

Tomando los datos de la tabla 6, el cálculo de la prioridad del atributo Funcionalidad quedaría:

$\bar{X} = (\sum X_i * f_i) / f_i$

$\bar{X} = 17 / 4$

$\bar{X} = 4,25$

Priorización a nivel de sistema

Una vez establecida la prioridad de los atributos de calidad por cada subsistema, es posible determinar la prioridad general del sistema, tomando para el cálculo de esta dos grupos de valores:

1. Los valores de prioridad calculados para cada subsistema.
2. Los valores de prioridad de atributos y subatributos en base a los requisitos no funcionales generales del sistema.

El segundo grupo de valores se calcula análogamente al proceso de cálculo de prioridades por subsistema, o sea, haciendo uso de la media aritmética.

Una vez calculados ambos grupos, los valores se agrupan como en la tabla 7.

Tabla 7. Valores generales de los atributos de calidad

Sistemas	Funcionalidad	Confiabilidad	Usabilidad	Eficiencia	Mantenibilidad	Portabilidad
Subsistema 1	4,25	3,30	3,90	3,70	3,60	3,50
Subsistema 2	3,10	3,90	3,80	3,70	3,30	3,40
Subsistema 3	4,00	3,90	3,60	3,80	2,80	3,50
Sistema	4,40	3,70	3,50	3,50	3,00	3,20

La tabla 7 muestra un ejemplo de la agrupación de valores para atributos, pero de esta misma manera habrá de hacerse para los subatributos. Para determinar la prioridad final del sistema, tanto de atributos y subatributos se emplea igualmente la media aritmética, teniéndose siempre en cuenta que, a mayor valor de media obtenido, mayor será la prioridad.

Tomando como base los valores de la tabla 7, es posible calcular la media aritmética de cada atributo (figura 10) y de ahí determinar las prioridades de cada uno.

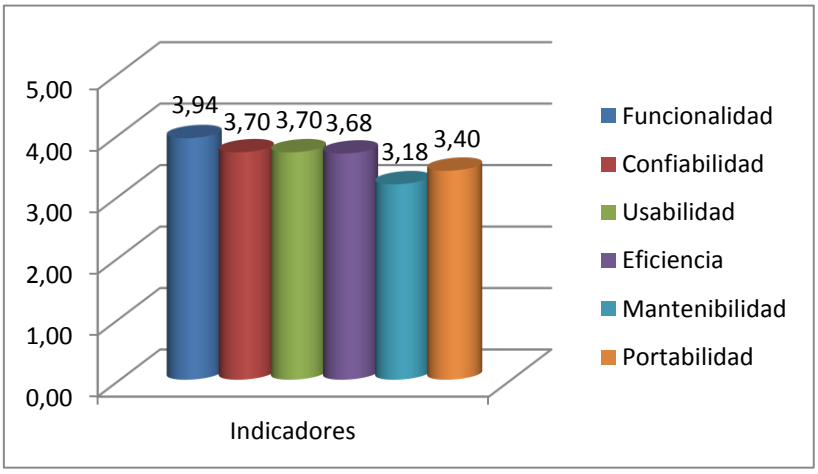


Figura 10. Media aritmética de los atributos para el sistema

En aquellos casos en que los valores obtenidos coincidan para más de un atributo (véase Confiabilidad y Usabilidad en la figura 10), se propone emplear la *desviación estándar*, como criterio estadístico que permite acortar la brecha de variaciones entre los valores de prioridad para un mismo atributo a calcular. Haciendo uso de este criterio, será posible distinguir cuál de los indicadores con igual valor de media debe tener mayor prioridad, en este caso sería el de menor desviación de sus valores de origen.

La d*esviación estándar* se utiliza cuando hay varios resultados posibles y estos están dispersos, lo cual acarrea inseguridad en el resultado final. Mientras más concentrados estén los resultados, habrá más confianza en el resultado. Lo anterior se traduce: a menor valor calculado entre una serie de elementos con valores dispersos, mayor será la probabilidad de ocurrencia de ese elemento.

Siendo la *desviación estándar* igual a la raíz cuadrada positiva de la *varianza*, se calcula entonces el valor de la última mediante la fórmula: $\mathbf{S^2 = \sum^n (X_i - \dot{X})^2 / n}$, donde:

$\mathbf{S^2}$ = Varianza

$\boldsymbol{X_i}$ = Valor del atributo en cada caso.

$\mathbf{\dot{X}}$ = Valor de la media del conjunto de valores asociados al subatributo/atributo.

$\boldsymbol{n}$ = cantidad total de valores analizados.

Si se toman los valores de muestra de la tabla 7 y se calcula la *varianza* para los atributos confiabilidad y usabilidad, quedaría:

Confiabilidad

$\mathbf{S^2} = ((3,3 – 3,7)^2 + (3, 9 – 3,7)^2 + (3,9 – 3,7)^2 + (3,7 – 3,7)^2) / 4$

$\mathbf{S^2} = (0,16 + 0,04 + 0,04 + 0) / 4 = 0,06$

$\mathbf{S} = \sqrt{\mathbf{S^2}} = \sqrt{0,06} = 0,24$

Usabilidad

$\mathbf{S^2} = ((3,9 – 3,7)^2 + (3,8 – 3,7)^2 + (3,6 – 3,7)^2 + (3,5 – 3,7)^2) / 4$

$\mathbf{S^2} = (0,04 + 0,01 + 0,01 + 0,04) / 4 = 0,025$

$\mathbf{S} = \sqrt{\mathbf{S^2}} = \sqrt{0,025} = 0,16$

Los cálculos realizados arrojan un menor valor para el atributo Usabilidad, por lo cual la prioridad mayor en este caso deberá ser para ese indicador. La priorización entonces quedaría (recuérdese que a mayor valor, mayor prioridad):

6. Funcionalidad
5. Usabilidad
4. Confiabilidad
3. Eficiencia
2. Portabilidad
1. Mantenibilidad

De esta manera ha sido posible combinar criterios estadísticos para determinar la prioridad que finalmente tendrán los atributos y subatributos de calidad para cada subsistema y para el sistema como un todo. Con esta propuesta, se armonizan las solicitudes y expectativas de calidad de arquitectos e interesados, con características de los grandes sistemas de software: modulares e integrados.

Sin embargo, no basta con eliminar los conflictos entre intereses y características de sistema, pues siguen quedando fuera de análisis los conflictos propios que se generan entre atributos.

2.3.2. Relaciones entre atributos y su impacto en la calidad

El presente trabajo busca guiar a los evaluadores de QUASAR en la consideración y revisión de las decisiones arquitectónicas tomadas por los arquitectos del sistema para lograr “la coexistencia pacífica” y complacencia (según orden de prioridad) de los atributos de calidad del sistema.

En la búsqueda por combinar la prioridad y el análisis de conflictos, y finalmente obtener un análisis que pueda ser útil a los arquitectos en la revisión de sus decisiones, ya sea en el proceso de desarrollo o como salida de su proceso de evaluación arquitectónica, se propone entonces la utilización de un algoritmo de grafos.

Análisis del algoritmo

El objetivo de este algoritmo es encontrar las variantes factibles y la óptima en cuanto a decisiones arquitectónicas, que permitan lograr las prioridades de los indicadores de calidad, sin generar conflictos entre ellos.

Las entradas necesarias para el algoritmo son:

1. La matriz de relaciones.
2. La lista de prioridades de atributos (mayor valor = mayor prioridad).

Tomando una matriz de relaciones entre atributos (véase epígrafe 1.1.2) como punto de partida, se construye un grafo dirigido y ponderado que tiene tantos nodos como atributos contempla la matriz.

Las aristas del grafo y sus direcciones se construyen a partir de las filas, o sea, el origen lo determina la fila y el fin, la columna. Los pesos de cada camino se establecen en función de las formas de afectación que se valoran en la matriz, tal y como se muestra en la tabla 8.

Tabla 8. Transformación de los valores en las celdas de la matriz por pesos en el grafo

Valor en la matriz	Peso en el grafo
+	1
-	-1
Sin valor	0

Para guiar los posteriores pasos de la propuesta se emplea el grafo de ejemplo de la figura 11.

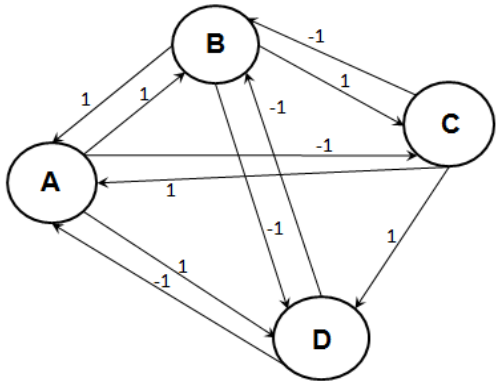


Figura 11. Grafo de relación entre atributos genéricos A, B, C y D

Una vez construido el grafo se procede a recorrerlo para obtener la mayor *importancia* posible de cada atributo según orden de prioridad. A los efectos de esta investigación la *importancia*

de un atributo se considera una combinación de la cantidad de atributos que afecta y que lo afectan y la importancia de estos, matemáticamente representada quedaría:

$$I_x = C_x * \sum_{i=1}^{C_x} R_i * P_i + A_x * \sum_{i=1}^{A_x} R_i * P_i$$

Donde:

I: Importancia de un nodo.

C_x: Cantidad de nodos que afecta el nodo x.

A_x: Cantidad de nodos que afectan al nodo x.

R: Prioridad de un nodo.

P: Peso.

x: Nodo.

Realizando un análisis sobre la fórmula de *importancia* se podría decir que predominan dos criterios:

1. Un atributo es más importante mientras más nodos afecta y lo afectan positivamente.
2. Un atributo es más importante mientras más prioritarios son los nodos que afecta y lo afectan positivamente.

Siguiendo el ejemplo de la figura 11, y tomando prioridades descendentes en la lista alfabética (A=4…D=1), la *importancia* de cada nodo, para el estado inicial del grafo, quedaría:

$$I_A = C_A * \big((R_B * P_B) + (R_C * P_C) + (R_D * P_D)\big) + A_A * ((R_B * P_B) + (R_C * P_C) + (R_D * P_D))$$

$$I_A = 3 * \big((1 * 3) + (-1 * 2) + (1 * 1)\big) + 3 * ((1 * 3) + (1 * 2) + (-1 * 1))$$

$$I_A = 6 + 12 = 18$$

Análogamente se calculan el resto de los valores de *importancia* para cada nodo.

Tomando en cuenta estos elementos, se propone utilizar un algoritmo *Vuelta atrás* (del inglés *backtracking)* que permita encontrar el caso óptimo para el conjunto de valores del grafo. De manera general el diseño y basamento del algoritmo quedaría:

Conjunto de candidatos: Corresponde a las *n* entradas del problema.

En este caso sería la matriz de relaciones entre los atributos de calidad establecidos en la norma ISO 9126-1, donde n = tamaño de la matriz.

Para el caso de la figura 11, las entradas corresponderían a los atributos genéricos A, B, C y

D.

Función de selección: Determina el candidato idóneo para formar la solución.

Esta función debe calcular todos los valores posibles de *importancia* para cada atributo de la matriz. Teniendo en cuenta que es un algoritmo *Vuelta atrás*, todos los valores son seleccionados para luego determinar la factibilidad de cada conjunto de valores.

La determinación de todos los valores en este paso llevará implícito el mantener o no determinadas aristas del grafo en cada iteración de cálculo, o sea, los valores serán calculados por cada grafo que se genere a partir del original, donde siempre el grafo que se cree tendrá menos aristas que su predecesor. La cantidad de grafos generados a partir del original en cada paso equivale a la cantidad de combinaciones sin repetición y se puede calcular mediante la siguiente:

$$\sum_{r=0}^{r=n} \frac{n!}{r!(n-r)!}$$

Donde *n* es la cantidad de aristas del grafo original y *r* es la cantidad en cada caso restante.

Siguiendo el ejemplo de la figura 11, quedaría: *n* = 11 y *r* = {0, 2, 3….11}.

Aplicando la fómula anterior y los valores de *n* y *r* anteriores, la cantidad de grupos de valores seleccionados es 2048.

Este paso puede ser representado como un algoritmo de ramificación, cuya representación visual se muestra en la figura 12.

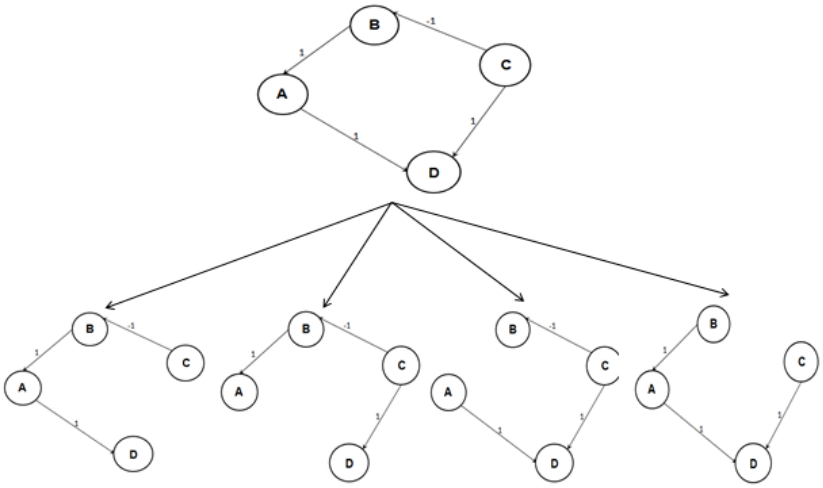


Figura 12. Representación del algoritmo de ramificación

En la figura12 se muestra el primer nivel del árbol que siempre tendrá tantos grafos como aristas tenga el original.

Función de factibilidad: Determina si es factible añadir un candidato a una solución.

Para esta función se tiene todo el conjunto de valores calculados en la función de selección, y se determina cuáles grupos de valores (nótese aquí que cada grupo está conformado por la *importancia* calculada para cada indicador en un estado dado del grafo) cumplen las siguientes condiciones:

1. $I_x > I_y$ cuando $R_x > R_y$

Si se toman los 2048 grupos de valores seleccionados y se le realiza el análisis de factibilidad, quedan un total de 426 casos factibles. Para este ejemplo uno de los casos factibles sería:

1. Un grafo cuyos valores de *importancia* son:
   $I_A = 20$, $I_B = 16$, $I_C = 9$, $I_D = 8$

Función objetivo: Función a optimizar.

Una vez obtenidos los grupos de valores factibles entonces se pasa a identificar el caso óptimo. Para los efectos de esta investigación se propone como caso óptimo aquel que cumpla las siguientes condiciones:

1. Menor desviación de los valores del grupo.
2. Mayores valores posibles por cada atributo.

Otra lectura de las condiciones anteriores sería: el caso óptimo es aquel que logra alcanzar sus mayores *importancias* en la totalidad del conjunto, sin diferencias significativas entre ellos. Para el logro de estos elementos se procede, en ese mismo orden, a encontrar los candidatos de la solución. Para ello, tomando como ejemplo el grafo de la figura 11, se tendrían 305 casos factibles. En este caso lo primero que se haría sería calcular la media de *desviación estándar* y la media de importancia de los 305 conjuntos.

La *desviación estándar* para cada grupo de valores factibles no es más que la raíz cuadrada de la v*arianza,* cuya fórmula es: $\mathbf{S^2 = \sum^n (X_i - \dot{X})^2 / n}$, donde:

$\boldsymbol{X_i}$ = Valor de *importancia* de 1 atributo en 1 grupo de casos factibles.

$\mathbf{\dot{X}}$ = Media de los valores de *importancia* del grupo.

$\boldsymbol{n}$ = Cantidad de atributos del grupo.

Se escogerían seguidamente aquellos casos factibles que sobrepasen la media y estén por debajo de la desviación. Es para estos grupos antes filtrados que se selecciona el de mayor primer valor, siendo ese el caso óptimo. Si existieran más de un conjunto con igual valor

máximo, se tomaría de ellos el de mayor valor de desviación. Para el ejemplo que se viene desarrollando el valor máximo de *importancia* de los casos factibles por debajo de la media de desviación es 20, y existen 6 casos en esta situación, por lo cual se escoge como caso óptimo el de mayor desviación.

El conjunto de valores de *importancia* de dicho caso es: $I_A = 20$, $I_B = 19$, $I_C = 9$, $I_D = 5$ y el grafo resultante sería como el de la figura 13.

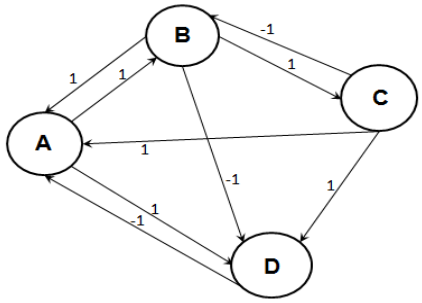


Figura 13. Caminos del caso óptimo

De este resultado final del algoritmo se puede concluir que el caso óptimo obtenido es aquel que tiende a mantener relaciones positivas en la matriz de relaciones, o sea, que pretende orientar al arquitecto en tomar decisiones que favorezcan mutuamente a los atributos que por naturaleza se entrelazan en soluciones arquitectónicas. Sin embargo, al introducir los criterios asociados a la desviación y la media se está favoreciendo que se tomen los mayores valores y conjuntos menos desviados pero eliminando la menor cantidad de aristas. Por tanto, se estaría obteniendo un caso óptimo no muy difícil de lograr.

En cualquier caso, siendo posible que esto no siempre logre, se recomienda utilizar el algoritmo para el análisis del resto de los casos factibles.

2.4. Propuesta y predefiniciones de QUASAR

La propuesta que se realiza tiene fundamento en deficiencias detectadas y además reconocidas por el propio autor del método QUASAR. Es fundamental, en cualquier caso, conocer dónde hacer uso de las nuevas variaciones que se proponen para la suplencia de las carencias mencionadas.

La primera deficiencia enunciada por los autores de QUASAR reza así:

"El método QUASAR hace énfasis en la evaluación de arquitecturas de sistemas, centrado en la evaluación de arquitecturas de subsistema en términos de la evaluación de las

arquitecturas de sus sub-subsistemas[12]*, y así sucesivamente. Al evaluar las arquitecturas de los subsistemas individuales en el mismo nivel de la jerarquía del sistema, se hace fácil pasar por alto el "bosque entero de los árboles individuales."* [47]

Para la resolución de esta deficiencia se propone lo planteado en el epígrafe 2.2 de la presente investigación.

Impacto en las fases y tareas

Enmarcado en el contexto de QUASAR la propuesta tendría incidencias en las siguientes fases definidas por el método [43]:

1. Fase inicial de evaluación de la arquitectura de sistema

    1.1. Fase de revisión de los requisitos del subsistema

Esta fase se repite para cada subsistema cuya arquitectura está siendo evaluada. Se asegura de que los requisitos de calidad asociados están debidamente asignados y especificados. También asegura que el equipo de arquitectura está listo para desarrollar sus casos de calidad.

    1.2. Fase de evaluación de la arquitectura del subistema

Esta fase se repite para cada subsistema cuya arquitectura está siendo evaluada. Durante esta fase, el equipo de arquitectura de subsistema presenta sus casos de calidad al equipo de evaluación.

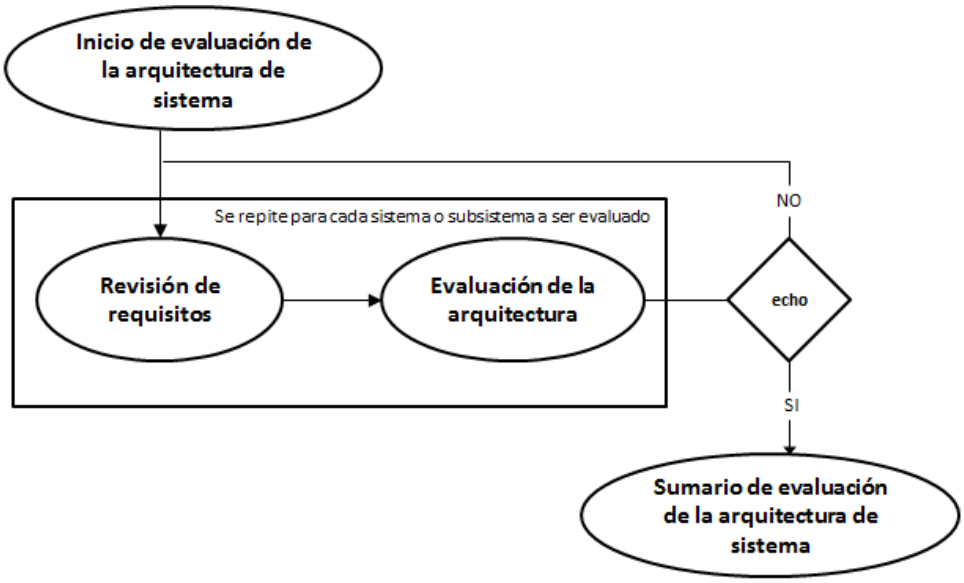


Figura 14. Representación de las fases propuestas por QUASAR [56]

[12] Componentes para el caso del sistema Cedrux.

Atendiendo a la representación de las fases en la figura 14, la propuesta tiene su mayor incidencia en la fase de Revisión de los requisitos, puesto que permite hacer uso del patrón definido en 2.2.1 para asegurar tener requisitos de calidad que sean medibles y correctamente descritos, además de permitir la realización de separaciones lógicas de requisitos en las evaluaciones del sistema y sus subsistemas. Acorde a [47], la utilización de dicho patrón incide fundamentalmente en la actividad Entrevista de Evaluación. Lo planteado en 2.2 también incluye variaciones en la ejecución de la evaluación para el sistema, puesto que no solo se revisarían los requisitos generales del sistema sino las dimensiones no funcionales (asociadas a atributos de calidad) de los requisitos funcionales de integración. Esto permitiría evaluar la calidad integral del sistema, tomando como elemento fundamental, su composición a partir de subsistemas.

Lo anteriormente planteado constituiría una extensión de la caracterización de las tareas de esta fase, donde se incluya la especificación para los casos en que la iteración de evaluación incluya integración de subsistemas o sub-subsistemas, casos para los cuales se necesitaría un caso de calidad de sistema, que permita medir la calidad de las integraciones.

Impacto en los artefactos

Para el desarrollo de una de las actividades de esta fase inicial se hace uso de varios artefactos, dentro de los que se cuenta la Lista de chequeo de requisitos *(figura 15)*. Esta lista de chequeo, para el caso que se expone, debe incluir en la sección relativa a la verificación de los requisitos de calidad, una pregunta referente al cumplimiento del patrón establecido para la descripción de los mismos y sobre la transformación de restricciones de diseño y RNF de otras categorías en requisitos de calidad del sistema.

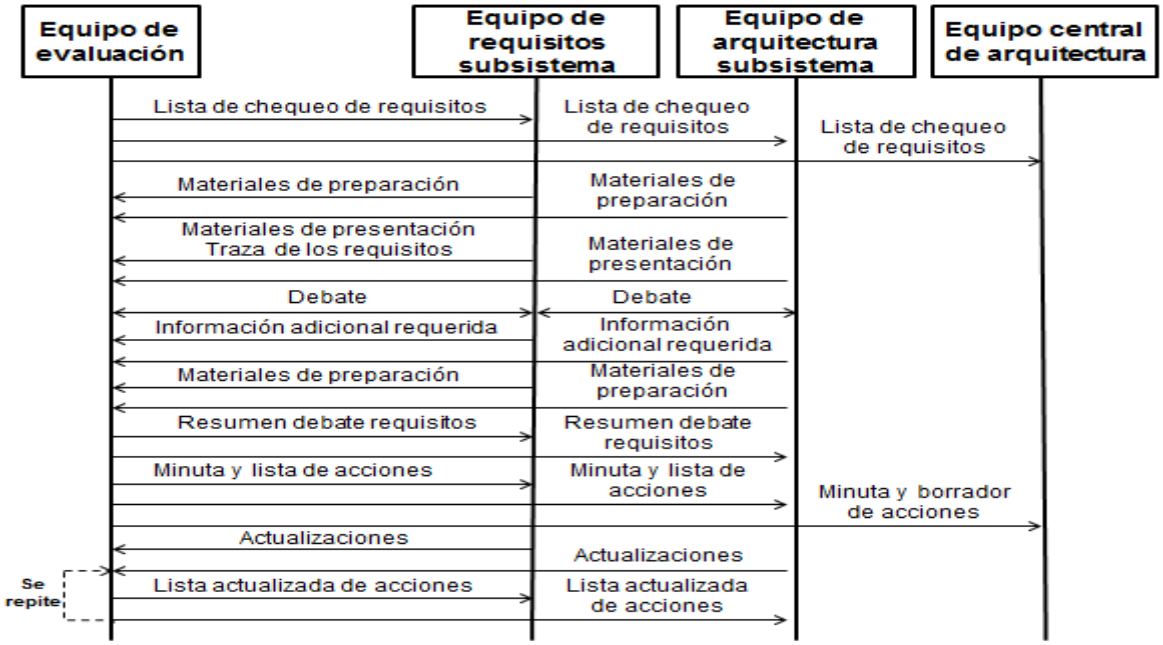


Figura 15. Artefactos de la actividad Entrevista de Evaluación en la fase Revisión de los requisitos [47]

Impacto en los roles y responsabilidades

En el caso de los equipos definidos previamente por QUASAR (figura 16) no sufren modificaciones en sus responsabilidades sino que la propuesta de esta investigación contribuye al mejor desempeño de sus funciones. Por ejemplo, con lo propuesto en 2.2.1 se ayuda a los equipos de requisitos a llevar a cabo las siguientes:

1. Garantizar que todos los requisitos (especialmente los arquitectónicamente significativos) sean cohesionados, completos, consistentes, correctos, actualizados, observables externamente, factibles, obligatorios, relevantes, orientados a los intereses de las partes involucradas, no ambiguos, y puedan ser validados y verificados.
2. Colaborar con el equipo de arquitectura de subsistema para asegurar que todos los requisitos arquitectónicamente significativos están debidamente:
   - ✓ Determinados
   - ✓ Identificados
   - ✓ Especificados con suficiente detalle para proporcionar orientación adecuada

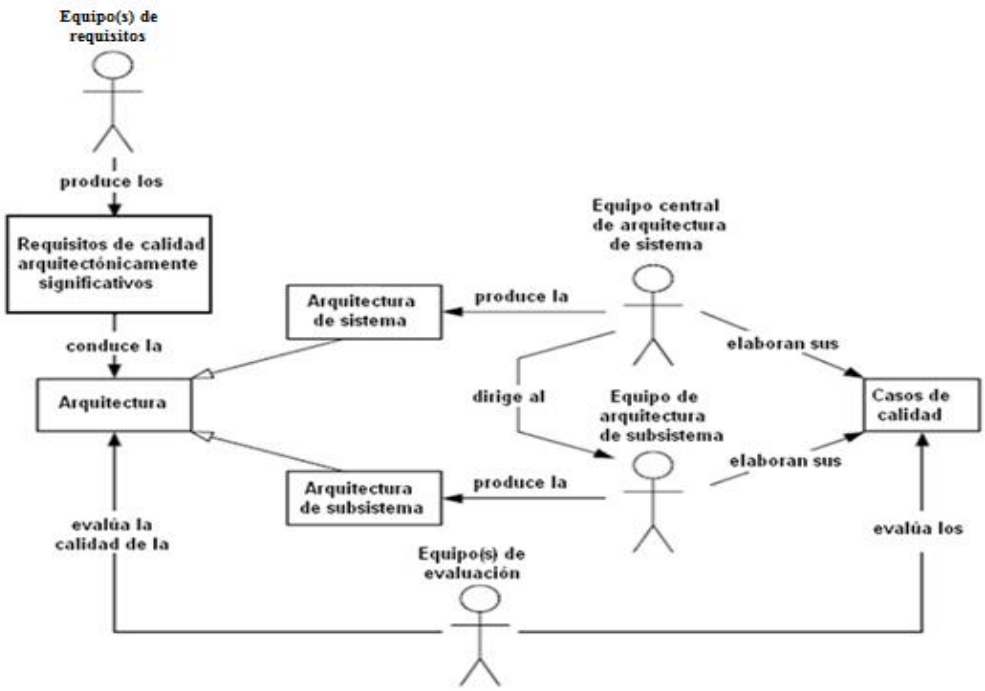


Figura 16. Equipos y sus interacciones [46]

Una segunda deficiencia enunciada por los autores de QUASAR es [47]:

"La versión actual del método QUASAR hace hincapié en la evaluación del soporte que brinda la arquitectura del sistema a un conjunto cohesionado de objetivos de calidad y requisitos mediante un factor de calidad individual. Esto se debe a que los casos de calidad consisten en demandas, argumentos y evidencias relacionadas con un factor o sub-factor de calidad individual. Sin embargo, un arquitecto de sistemas debe hacer balances ingenieriles entre los diferentes factores de calidad."

Impacto en las fases y tareas

Para la fase de Revisión de los requisitos se llevan a cabo, entre otros, pasos tales como:

Internamente revisar los objetivos de calidad y requisitos

En esta actividad los equipos de requisitos y arquitectura se aseguran de que los requisitos estén debidamente especificados, agrupados por subsistema y priorizados.

Este último elemento es también garantizado por la presente propuesta, mediante lo plasmado en el epígrafe 2.3.1 para el cálculo de priorización de atributos a partir de la clasificación de los requisitos de calidad del sistema.

Acorde a las premisas del método, la priorización sirve como base para la implementación, pero fundamentalmente para determinar la importancia de aquellos RNF que son

arquitectónicamente significativos. Los evaluadores deben analizar el cumplimiento de dichas prioridades mediante los argumentos arquitectónicos brindados para su logro.

Como parte de las fases de QUASAR se encuentra también la de Evaluación de la Arquitectura (figura 14).

Evaluación de la Arquitectura

Esta fase se repite por cada subsistema cuya arquitectura es objeto de evaluación. Durante ella, los arquitectos de cada subsistema presentan sus casos de calidad a los equipos de evaluación [47].

Es de vital importancia para este paso incluir en los argumentos de cada caso de calidad la justificación de una decisión arquitectónica desde el punto de vista de su impacto sobre otros atributos o sub-atributos de calidad relacionados al del caso en cuestión. Para ello se propone hacer uso de lo plasmado en el epígrafe 2.3.2, por ejemplo:

En un caso de calidad de Rendimiento , el argumento X refiere cómo permite apoyar la demanda Y del caso de calidad en cuestión. Sin embargo, este argumento debe hacer alusión a que la decisión impacta negativa o positivamente sobre el atributo Operabilidad acorde a lo permitido según caso factible u óptimo obtenido por el algoritmo de 2.3.2.

Asimismo, en la fase Sumario de la evaluavión de la arquitectura de sistema cuyos objetivos fundamentales son [47]:

1. Recoger los resultados de todas las evaluaciones anteriores de la arquitectura de los subsistemas y documentar la evaluación de la calidad general resultante de la arquitectura del sistema.
2. Realizar un resumen final de lo que fue bien y qué podría mejorarse en relación con el proceso de evaluación de la calidad de la arquitectura del sistema.

En este fase donde se dan los resultados es muy importante entregar a los equipos de requisitos y arquitectura la información en cuanto a lo que fue mal de una manera productiva, o sea, indicar qué estuvo mal y posibles causas y vías de solución. De nada sirve que estos resultados sean cualitativos o cuantitativos sin “interpretación”, por ejemplo: el atributo Rendimiento dio un resultado de Mal en su revisión.

Para ello la propuesta del epígrafe 2.3.2 indica al equipo de evaluadores los caminos que deben tomar las decisiones para el equilibrio de los atributos, de manera que, si el equipo evaluador al revisar los casos de calidad detecta problema en un atributo, pueda verificar si la causa tiene que ver con la interferencia de alguna decisión mal tomada para favorecer otro. Si fuera este el caso, el resultado de esa evaluación es mucho más precisa para el equipo de arquitectura, cuya tarea sería variar la decisión.

Impacto en los roles y responsabilidades

En el caso del equipo de arquitectura de subsistema definido previamente por QUASAR (figura 16), no sufre modificaciones en sus responsabilidades sino que la propuesta de esta investigación contribuye al mejor desenvolvimiento de sus funciones. Lo propuesto en 2.3.2 implica para el equipo seguir llevando a cabo las siguientes:

1. Preparar y presentar los casos de calidad al equipo de evaluación para convencerlos de que el subsistema adecuadamente soporta sus requisitos de calidad identificados y derivados.
2. Responder las preguntas de los evaluadores con respecto a sus decisiones arquitectónicas.

Sin embargo, en ambos casos, el equipo debe incluir el enfoque de relación entre atributos y demostrar, ya sea mediante la presentación de los casos de calidad o mediante las respuestas a los evaluadores, que la arquitectura propuesta cumple las prioridades definidas para los atributos de calidad del subsistema y en adecuado equilibrio de los mismos, sustentándose en lo propuesto en 2.3.1 y 2.3.2.

2.5. Conclusiones parciales

Tras la realización de una propuesta de variaciones al método QUASAR es posible concluir que:

1. Las variaciones propuestas permiten a QUASAR determinar la calidad integral de un producto de software modular e integrado mediante el uso de un caso de calidad de sistema.
2. La estadística es aplicable en el contexto de determinación de las prioridades de atributos de calidad.
3. El empleo del algoritmo propuesto permite evaluar y encaminar las decisiones arquitectónicas en función de prioridades y relaciones entre atributos para evitar los conflictos entre estos, supliendo así la deficiencia de QUASAR en este sentido.

CAPÍTULO 3. ANÁLISIS DE RESULTADOS

En este capítulo se demuestra la aplicabilidad del método QUASAR, con las variaciones propuestas, en el contexto de desarrollo de los productos asociados a Cedrux. Además, se verifica el comportamiento de la comprobabilidad de la calidad interna ante la aplicación de QUASAR modificado. Finalmente se analizan los resultados haciendo uso de los test de Mann Whitney y Wilcoxon.

3.1. Caracterización de la muestra

La aplicación de la propuesta se realizará contando con Cedrux y sus *productos de base*[13] como población, para un total de 12. La selección de la muestra se realizó de forma no probabilística, obteniendo aquellos productos que por su estado actual permiten una recolección de datos fiables y para los cuales las características arquitectónicas son muy similares y acordes a las planteadas en la investigación, para un total de 9 productos (tabla 9), equivalentes al 75 % de la población.

Tabla 9. Muestra seleccionada de productos de software

Nombre del producto	Siglas	Estado actual	Cantidad de subsistemas que integra
Planificación Empresarial y Presupuestada	PEP	Desarrollo	3
Auditoría	INV	Pruebas de aceptación	3
Contabilidad	CONT	Pruebas de aceptación	3
Costos y Procesos	COST	Pruebas de aceptación	4
Finanzas	FIN	Pruebas de aceptación	5
Capital Humano	CH	Pruebas de aceptación	5
Inventario	EC	Pruebas de aceptación	6
Facturación	AUD	Pruebas de liberación interna	7
Cedrux	CED	Pruebas de aceptación	9

Todos los productos de la muestra seleccionada presentan un conjunto de características similares desde el punto de vista de la construcción de su arquitectura, estas son:

1. Fueron construidos bajo el mismo modelo de desarrollo.
2. Son modulares e integrados.
3. Poseen la misma línea base de la arquitectura (patrones arquitectónicos y protocolos

[13] Productos a partir de los cuales se construye Cedrux. Desde la perspectiva de Cedrux dichos productos son subsistemas integrados, pero por si mismos constituyen también productos modulares e integrados.

de integración).

4. Poseen la misma base tecnológica.

3.2. Diseño del experimento

Para la validación de la investigación se hará uso de procesamiento estadístico mediante el uso de los test Mann Whitney y Wilcoxon. La ejecución de las pruebas se hará como muestra la figura 17.

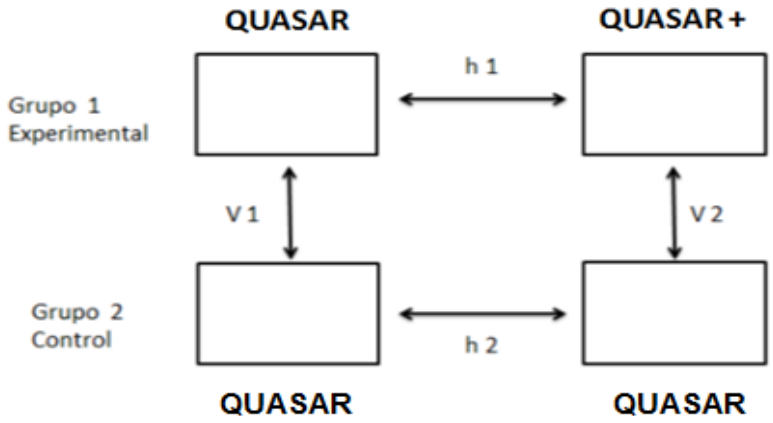


Figura 17. Diseño del experimento

El test Mann Whitney será usado para la ejecución de V1 y V2 en el análisis de dos muestras independientes. Por su parte, el test Wilcoxon será empleado para la ejecución de h1 y h2 en el análisis de dos muestras apareadas.

La selección de los grupos experimentales se realizó tomando en cuenta la cantidad de subsistemas integrados en cada producto, de manera que cada grupo tuviera productos grandes (5 o más subsistemas) y productos pequeños (menos de 5 subsistemas) Finalmente el grupo experimental (1) tiene 5 productos y el grupo control (2) tiene un total de 4 productos (figura 18).

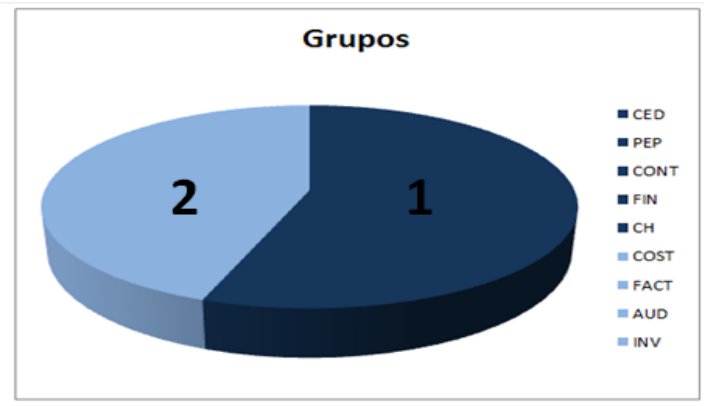


Figura 18. Representación de productos en cada grupo

3.3. Análisis de indicadores

Para el análisis de la capacidad de prueba tras las variaciones propuestas a QUASAR, se identificaron dos variables y sus indicadores respectivos. El desglose de los mismos se muestra en la tabla 10.

Tabla 10. Operacionalización de variables de la investigación

Variable	Dimensión	Indicador	Se propone medir	Medición (fórmula)	Interpretación del valor
Método de evaluación	Aplicabilidad	Nivel de soporte a las actividades	¿Presenta el método mecanismos de soporte para sus actividades?	No aplica	Si o No
		Guía para el análisis causal a partir de los resultados	¿Se auxilia el método de alguna guía para analizar posibles causas de problemas de cumplimiento de atributos?		Si o No
		Cubrimiento de las necesidades de priorización y solución de conflictos	¿El método tiene en cuenta la priorización de atributos y los conflictos de atributos de calidad?		Si o No

Capacidad de prueba	Arquitectura	Cubrimiento de requisitos no funcionales de atributos de calidad (CRNFAC)	¿Es comprobable el cubrimiento por la arquitectura de los RNF de atributos de calidad?	**X = A/B** A: Requisitos comprobables B: Total de requisitos.	**0 <= X <= 1** A mayor cercanía al 1 mejor.
		Capacidad de análisis de conflictos (CAC)	¿Es posible analizar el impacto de las decisiones arquitectónicas en más de un atributo?	**X** = A/3 A: Cantidad de análisis posibles.	A mayor cercanía al 1 mejor.
		Nivel de priorización (NP)	¿Cuál es el nivel de priorización de los atributos de calidad?	**X = A/8** A: Cantidad de combinaciones elemento-dimensión a priorizar.	A mayor cercanía al 1 mejor.

3.4. Datos de la muestra

Para la obtención de los datos se realizó un análisis de cada uno de los indicadores en la muestra seleccionada.

1. Cubrimiento de requisitos no funcionales de atributos de calidad

Para el caso de este indicador se analizaron los requisitos no funcionales de atributos de calidad de los productos de la muestra. El análisis se realizó a partir de la revisión de la documentación en aras de determinar:

- ✓ La cantidad de RNF de atributos de calidad en cada subsistema de cada producto.
- ✓ La cantidad de RNF de atributos de calidad generales de cada producto, en base a requisitos comunes entre subsistemas de base.

Por ejemplo, el producto Auditoría se compone de 3 subsistemas: Estructura y composición (EC), Configuración general (CONF) y Auditoría (AUD). Una muestra del análisis realizado con los requisitos de este producto se evidencia en la tabla 11.

Tabla 11. Análisis de RNF de atributos de calidad en producto Auditoría.

RNF_ AUD	RNF_ EC	RNF_ CONF	RNF_Comunes	RNF_Producto
53	53	60	43	80

Teniendo en cuenta que los requisitos comunes se restan a cada cantidad original de requisitos de los subsistemas y las cantidades resultantes se suman con los requisitos comunes, entonces la cantidad de requisitos del producto quedaría:

RNF_Producto = 10+10+17+43 = 80

Análogamente se realizó este análisis con cada producto.

Con todos los valores anteriormente calculados, el indicador Cubrimiento de requisitos no funcionales de atributos de calidad se podrá determinar conociendo cuántos requisitos del producto es posible evaluar mediante el uso de cada método.

2. Capacidad de análisis de conflictos

Tomando como referencia el término conflictos abordado en 1.1.2, y las características de la norma ISO/IEC 9126-1, es posible distinguir que el análisis de los conflictos entre atributos puede hacerse desde dos perspectivas:

1. Conflictos internos

   Son los conflictos que se generan entre subatributos de un mismo atributo.

2. Conflictos externos

   Estos conflictos se pueden clasificar en dos casos.

   2.1. Conflictos que se generan entre atributos.

   2.2. Conflictos que se generan entre subatributos de atributos diferentes.

Para extraer los datos de esta variable se modelan los casos de calidad por subsistema y sistema respectivamente en cada producto.

Haciendo uso del artefacto Matriz de soporte propuesto en QUASAR, donde se representa el resumen de la evaluación realizada a cada subsistema y/o sistema en función de colores (figura 19), es posible realizar análisis de los estados de cada indicador.

Evaluaciones	Funcionalidad	Idoneidad	Precisión	Seguridad	Interoperabilidad
Sistema PEP	Anaranjado	N/E	Rojo	Amarillo	N/E

Figura 19. Matriz de soporte para producto PEP

Un análisis sobre la figura 19 permite identificar que el color Anaranjado del atributo Funcionalidad está directamente determinado por los colores Rojo y Amarillo de sus respectivos subatributos, lo cual puede considerarse como conflictos internos. Sin embargo, si se pretendiera determinar el impacto que alguno de estos subatributos (relativos a Funcionalidad) estuviera teniendo sobre un subatributo clasificado en otro atributo (por ejemplo, Eficiencia) no es posible determinarlo mediante esta matriz.

Si se utiliza el algoritmo propuesto en 2.3.2 es posible conocer cuáles son las relaciones que deben existir entre los atributos para lograr un equilibrio en función de sus prioridades. Luego, solo se necesita buscar las decisiones (argumentos) favorecedoras de los atributos que afectan al atributo analizado, analizarlas respecto a lo obtenido tras la ejecución del algoritmo, para así determinar el impacto de dichas decisiones en el resultado final alcanzado.

3. Nivel de priorización

Teniendo en cuenta la fundamentación teórica realizada respecto a la evaluación de la arquitectura y las formas de desglose de atributos de calidad y su priorización respectiva, se pudo determinar que es posible:

- ✓ Priorizar 4 elementos desagregados a partir de atributos de calidad.
- ✓ Realizar la priorización desde dos dimensiones.

Teniendo lo anterior en cuenta se determinó que existen 8 niveles de priorización alcanzables, tal como muestra la tabla 12.

Tabla 12. Análisis de niveles de priorización

Elemento a priorizar	Dimensión	Nivel de priorización
Escenarios	Subsistema	1/8
	Sistema	1/8
Requisitos	Subsistema	1/8
	Sistema	1/8
Subatributos	Subsistema	1/8
	Sistema	1/8
Atributos	Subsistema	1/8
	Sistema	1/8

3.5. Análisis de resultados

Tomando como principio que la variable independiente de esta investigación toma valores de presencia (si) cuando se aplica QUASAR+[14] y ausencia (no) cuando se aplica QUASAR, entonces se somete a verificación el estado de la variable dependiente en ambos casos, tal y como se muestra en la figura 17.

Para todos los indicadores las mediciones se realizaron mediante el experimento de la figura 17. Los valores de los indicadores para ambos grupos se muestran en la tabla 13.

Tabla 13. Valores de los grupos aplicando QUASAR

Grupo	Producto	Cubrimiento de RNF de atributos de calidad (CRNFACI)	Nivel de priorización (NPI)	Capacidad de análisis de conflictos (CPCI)
1	CED	139/175	0/8	1/3
1	PEP	36/81	0/8	1/3
1	CONT	32/78	0/8	1/3
1	FIN	74/114	0/8	1/3
1	CH	72/113	0/8	1/3
2	COST	37/84	0/8	1/3
2	FACT	113/152	0/8	1/3
2	AUD	37/80	0/8	1/3
2	INV	96/135	0/8	1/3

[14] QUASAR con variaciones propuestas en el capítulo 2.

Realizando lo especificado en 3.2, se procedió a comprobar la no existencia de diferencias significativas entre el grupo experimental y el grupo de control durante la aplicación de QUASAR, (comprobación de V1) mediante el Test Mann Whitney para los indicadores Nivel de priorización y Capacidad de análisis de conflictos. Los valores resultantes se muestran en la figura 20.

Mann-Whitney Test

Ranks

	C	N	Mean Rank	Sum of Ranks
CRNFACI	1	5	4,70	23,50
	2	4	5,38	21,50
	Total	9		
NPI	1	5	5,00	25,00
	2	4	5,00	20,00
	Total	9		
CAC	1	5	5,00	25,00
	2	4	5,00	20,00
	Total	9		

Test Statistics[d]

			CRNFACI	NPI	CAC
Mann-Whitney U			8,500	10,000	10,000
Wilcoxon W			23,500	20,000	20,000
Z			-,369	,000	,000
Asymp. Sig. (2-tailed)			,712	1,000	1,000
Exact Sig. [2*(1-tailed Sig.)]			,730[a]	1,000[a]	1,000[a]
Monte Carlo Sig. (2-tailed)	Sig.		,778[b]		
	99% Confidence Interval	Lower Bound	,777		
		Upper Bound	,778		
Monte Carlo Sig. (1-tailed)	Sig.		,389[b]		
	99% Confidence Interval	Lower Bound	,388		
		Upper Bound	,389		
Exact Sig. (2-tailed)				1,000[c]	1,000[c]
Exact Sig. (1-tailed)				1,000[c]	1,000[c]
Point Probability				1,000[c]	1,000[c]

Figura 20. Test Mann Whitney en V1

Como se evidencia en la figura, las diferencias no fueron significativas, quedando demostrado así que ambos grupos al inicio del experimento se encontraban en igualdad de condiciones.

Los valores de los indicadores Nivel de priorización, Capacidad de análisis de conflictos y Cubrimiento de los RNF de atributos de calidad, luego de aplicado QUASAR+ en el Grupo 1 y aplicando nuevamente QUASAR en el Grupo 2, se muestran en la tabla 14.

Tabla 14. Valores de los grupos aplicando QUASAR y QUASAR+

Grupo	Producto	Cubrimiento de RNF	Nivel de	Capacidad de

		de atributos de calidad (CRNFACF)	priorización (NPF)	análisis de conflictos (CPCF)
1	CED	175/175	6/8	3/3
1	PEP	81/81	6/8	3/3
1	CONT	78/78	6/8	3/3
1	FIN	114/114	6/8	3/3
1	CH	113/113	6/8	3/3
2	COST	37/84	0/8	1/3
2	FACT	113/152	0/8	1/3
2	AUD	37/80	0/8	1/3
2	INV	96/135	0/8	1/3

Al realizar un análisis a partir de los valores de cada indicador en ambos estados (estado inicial: prueba con QUASAR, estado final: prueba con QUASAR Y QUASAR+) se demuestra el comportamiento de los valores en ambos grupos, esto equivale al test Wilcoxon en h1 y h2.

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
NPF - NPI	Negative Ranks	0[a]	,00	,00
	Positive Ranks	5[b]	3,00	15,00
	Ties	0[c]		
	Total	5		
CRNFACF - CRNFACI	Negative Ranks	0[d]	,00	,00
	Positive Ranks	5[e]	3,00	15,00
	Ties	0[f]		
	Total	5		
CAC - CAC	Negative Ranks	0[g]	,00	,00
	Positive Ranks	5[h]	3,00	15,00
	Ties	0[i]		
	Total	5		

Test Statistics[b,c]

			NPF - NPI	CRNFACF - CRNFACI	CAC - CAC
Z			-2,236[a]	-2,023[a]	-2,236[a]
Asymp. Sig. (2-tailed)			,025	,043	,025
Monte Carlo Sig. (2-tailed)	Sig.		,062	,063	,062
	99% Confidence Interval	Lower Bound	,062	,062	,062
		Upper Bound	,063	,063	,063
Monte Carlo Sig. (1-tailed)	Sig.		,031	,031	,031
	99% Confidence Interval	Lower Bound	,031	,031	,031
		Upper Bound	,032	,032	,032

Figura 21. Test Wilcoxon (h1) en grupo experimental

Como se evidencia en la figura 21, las diferencias en el grupo experimental resultaron significativas y para este grupo los valores finales de cada indicador fueron mayores que los iniciales, lo cual está acorde con lo reflejado en el epígrafe 3.3 respecto al análisis de dichos indicadores. Esto refleja un aumento de todos los indicadores para el caso del grupo experimental. Lo anterior evidencia que existe un aumento de la capacidad de prueba de la calidad interna del sistema ante las variaciones realizadas a QUASAR.

Para el caso de h2, tal como se muestra en la figura 22, las diferencias en el grupo de control no fueron significativas, lo cual evidencia que ante la inexistencia de variaciones a QUASAR se mantienen invariables los indicadores.

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
NPF - NPI	Negative Ranks	0[a]	,00	,00
	Positive Ranks	0[b]	,00	,00
	Ties	4[c]		
	Total	4		
CRNFACF - CRNFACI	Negative Ranks	0[d]	,00	,00
	Positive Ranks	0[e]	,00	,00
	Ties	4[f]		
	Total	4		
CAC - CAC	Negative Ranks	0[g]	,00	,00
	Positive Ranks	0[h]	,00	,00
	Ties	4[i]		
	Total	4		

Test Statistics[c]

	NPF - NPI	CRNFACF - CRNFACI	CAC - CAC
Z	,000[a]	,000[a]	,000[a]
Asymp. Sig. (2-tailed)	1,000	1,000	1,000
Exact Sig. (2-tailed)	1,000[b]	1,000[b]	1,000[b]
Exact Sig. (1-tailed)	1,000[b]	1,000[b]	1,000[b]
Point Probability	1,000[b]	1,000[b]	1,000[b]

Figura 22. Test Wilcoxon (h2) en grupo de control

Luego se procedió a comprobar la existencia de diferencias significativas entre el grupo experimental y el grupo de control durante la aplicación de QUASAR+ y QUASAR, respectivamente (comprobación de V2) mediante la aplicación del Test Mann Whitney (figura 23).

Mann-Whitney Test

Ranks

	C	N	Mean Rank	Sum of Ranks
CRNFACF	1	5	7,00	35,00
	2	4	2,50	10,00
	Total	9		
NPF	1	5	7,00	35,00
	2	4	2,50	10,00
	Total	9		
CAC	1	5	7,00	35,00
	2	4	2,50	10,00
	Total	9		

Test Statistics[c]

			CRNFACF	NPF	CAC
Mann-Whitney U			,000	,000	,000
Wilcoxon W			10,000	10,000	10,000
Z			-2,683	-2,828	-2,828
Asymp. Sig. (2-tailed)			,007	,005	,005
Exact Sig. [2*(1-tailed Sig.)]			,016[a]	,016[a]	,016[a]
Monte Carlo Sig. (2-tailed)	Sig.		,008[b]	,008[b]	,008[b]
	99% Confidence Interval	Lower Bound	,008	,008	,008
		Upper Bound	,008	,008	,008
Monte Carlo Sig. (1-tailed)	Sig.		,008[b]	,008[b]	,008[b]
	99% Confidence Interval	Lower Bound	,008	,008	,008
		Upper Bound	,008	,008	,008

Figura 23. Test Mann Whitney en V2

Tal y como se presenta en la figura 23, las diferencias en los valores de los indicadores analizados fueron significativas, reafirmándose así que las variaciones a QUASAR permiten obtener un aumento de la capacidad de prueba de la calidad interna el sistema.

Los resultados obtenidos con la aplicación de QUASAR y QUASAR+ para cada uno de los indicadores se muestran en las figuras 24, 25 y 26.

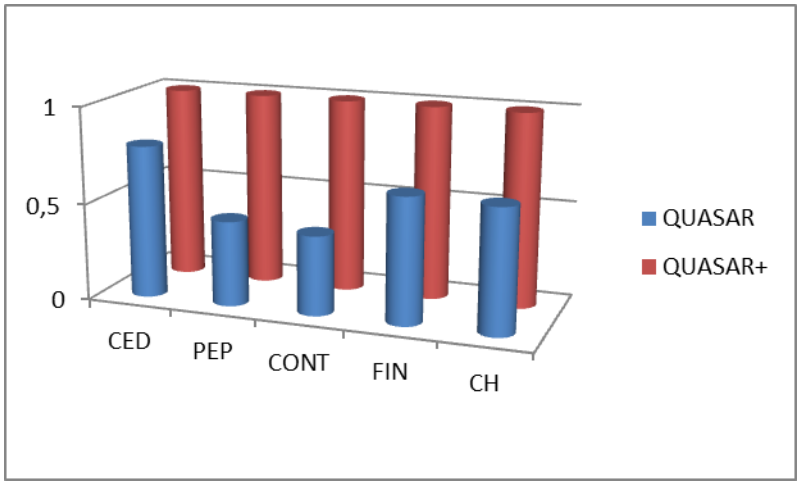


Figura 24. Valores del indicador CRNFAC en el experimento

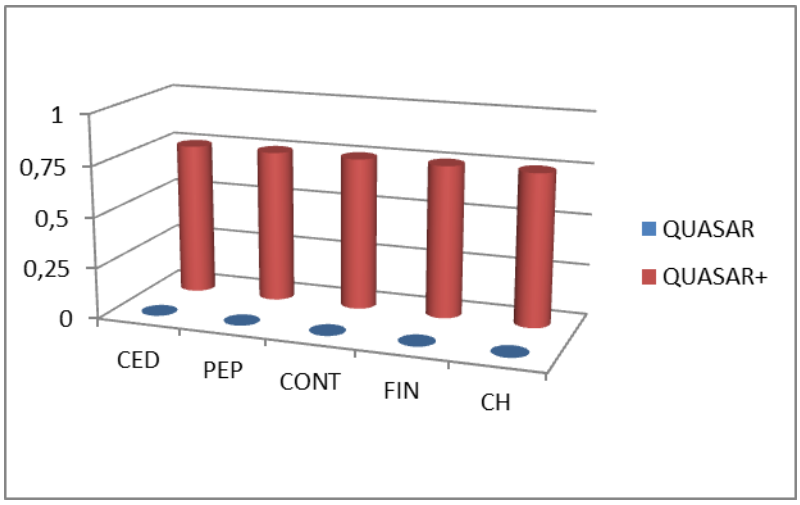


Figura 25. Valores del indicador NP en el experimento

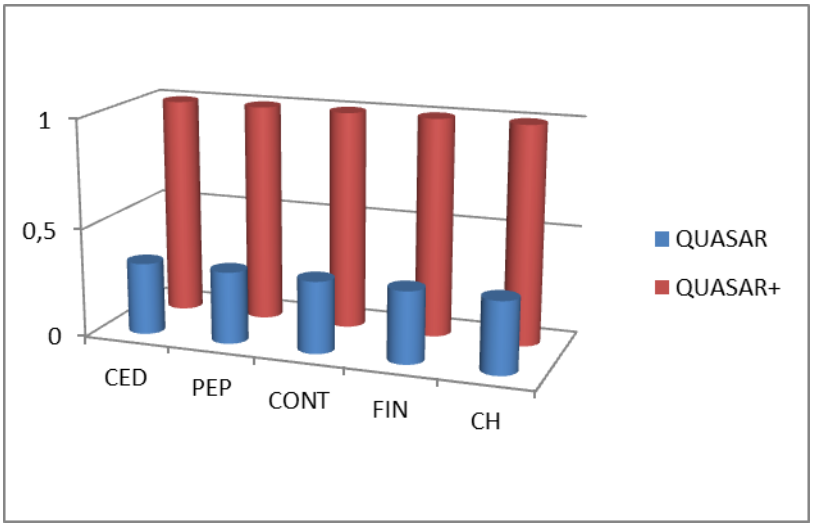


Figura 26. Valores del indicador NP en el experimento

Un análisis final sobre las gráficas presentadas en las figuras 25, 26 y 27 evidencian un aumento de todos los indicadores tras la aplicación de QUASAR+ y por tanto se puede afirmar el aumento de la capacidad de prueba de la calidad interna del sistema.

3.6. Conclusiones parciales

Luego de realizado un análisis de los indicadores asociados a la capacidad de prueba de la calidad interna del sistema Cedrux, habiendo aplicado QUASAR+, es posible concluir que:

1. El análisis estadístico de la variable capacidad de prueba en la muestra seleccionada,

arrojó resultados positivos para el grupo experimental y no mostró variación alguna para el grupo de control.

2. Las variaciones propuestas a QUASAR permiten aumentar la capacidad de prueba de la calidad interna del sistema Cedrux.

CONCLUSIONES

La realización de la presente investigación permite arribar a las siguientes conclusiones:

1. La evaluación de la arquitectura de software se auxilia hoy de diversos métodos y técnicas; dentro de ellos el método QUASAR presenta cualidades que le hacen adecuado para la evaluación de sistemas modulares integrados como los ERP.
2. La ejecución de un nuevo caso de calidad de sistema en QUASAR, permite realizar evaluaciones de la calidad integral de sistemas modulares, adecuando más este método al contexto de desarrollo de Cedrux.
3. La utilización de técnicas cuantitativas en la priorización de atributos de calidad y el análisis de interrelaciones, dotan al método QUASAR de herramientas ágiles para el análisis en el proceso de evaluación.
4. La experimentación de QUASAR+ en el entorno de desarrollo de Cedrux, permitió evidenciar un aumento de la capacidad de prueba de la calidad interna del sistema.

RECOMENDACIONES

Con el objetivo de dar continuidad a la presente investigación se recomienda:

1. Desarrollar una herramienta informática que soporte las bases de QUASAR, fundamentalmente referidas a la elaboración de los casos de calidad, y que a su vez incluya soporte para las variaciones aquí propuestas.
2. Utilizar para el desarrollo de la herramienta de soporte, lenguajes y tecnologías que permitan elevar la eficiencia, en aras de que sea más efectivo su uso.

BIBLIOGRAFÍA

1. **Benvenuto Vera, Ángelo.** *Implementación de sistemas ERP, su impacto en la gestión de la empresa e integración con otras TIC.* [pdf] Universidad de concepción : Capiv Review, Capiv Review, 2006.Vol.4.

2. **Salmerón Silvera, J. L. y López Vargas, C.** *Modelo bidimensional de riesgos del manteniemiento de sistemas integrados de gestión (ERP).* [pdf] Vigo, España : Investigaciones Europeas de Dirección y Economía de la Empresa., 2010. Vol.16. ISSN (Versión impresa): 1135-2523.

3. **Carreón Suarez del Real, María Cristina.** *Construcción de un catálogo de patrones de requisitos funcionales para ERP. Tesis de Máster.* [pdf] Cataluña : Universidad Politécnica de Cataluña, 25 de junio de 2008.

4. **Clements, Paul.** *A Survey of Architecture Description Languages.* Pittsburgh : Software Engineering Institute Carnegie Mellon University, 1996.

5. **de la Torre LLorente, et al.** *Guía de arquitectura N-Capas orientada al dominio con .NET 4.0.* BETA. España : Krasis Consulting, S.L., 2010. pág. 3. ISBN: 978-84-936696-3-8.

6. **Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel.** *Arquitecturas de software. Guía de estudio.* abril de 2004.

7. **Grimán Padua, Anna Cecilia.** *Evaluación Arquitectónica de la Calidad del software.* [pdf] s.l. : Universidad Simón Bolivar, Septiembre de 2005.

8. **Gómez, Omar.** *Evaluando la Arquitectura de Software. Parte 1. Panorama General.* México : Brainworx S.A. de C.V., enero-febrero de 2007.

9. **Berntsson Svensson, Richard, et al.** *Prioritization of Quality Requirements: State of Practice in Eleven Companies.* [pdf] s.l. : IEEE 19th International Requirements Engineering Conference, 2011.

10. **Hoffmann, Martin.** *Quality Evaluation of Software Architecture with Application to OpenH.323 Protocol. Master's Thesis in Information Technology.* [pdf] Jyväskylä : Department of Mathematical Information Technology. University of Jyväskylä., 3 de octubre de 2006.

11. **Panovski, Gregor.** *Master's Thesis. Product Software Quality.* [pdf] Eindhoven : Department of Mathematics and Computing Science.Technische Universiteit Eindhoven, febrero de 2008.

12. **Kim, Chanwook y Lee, Keun.** *Software Quality Model for Consumer Electronics Product.* [pdf] s.l. : 2009 Ninth International Conference on Quality Software, 2009.

13. **Marcos, José, et al.** *La norma ISO/IEC 25000 y el proyecto KEMIS para su automatización con software.* [pdf] Madrid, España : REICIS Revista Española de Innovación, Calidad e Ingeniería del Software, 2008. Vol.4. ISSN:1885-4486.

14. **Etaati, Leila , Sadi-Nezhad, Soheil y Makue, Ahmad .** *Using Fuzzy Group Analytical Network Process and ISO 9126 Quality Model in Software Selection: A case study in E-Learning Systems.* [pdf] Tehran, Irán : Journal of Applied Sciences, 2011. ISSN: 1812-5654.

15. **Galster, M., Eberlein, A. y Moussavi, M.** *Systematic selection of software architecture styles.* [pdf] Canada/ Emiratos Árabes Unidos : IET Software, 2010. ISSN 1751-8806.

16. **Comité Técnico y de Normalización NC/CTN 18 de Tecnologías de la Información.** *Norma Cubana. Ingeniería de software—Calidad del producto—Parte 1: Modelo de la calidad ISO/IEC 9126-1:2005.* [pdf] Ciudad de La Habana : Oficina Nacional de Normalización, 2005.

17. **Collabera, V. Gnanasekaran.** *Evaluating Application Architecture, Quantitatively.* [html] s.l. : The Architecture Journal, Marzo de 2010. Journal 24.

18. **Otero, Carlos E. , et al.** *A Quality-Based Requirement Prioritization Framework Using Binary Inputs.* [pdf] s.l. : Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, 2010.

19. **Real Academia Española.** Diccionario de la Lengua Española. *Real Academia Española.* [En línea] Real Academia Española (RAE), 2012. [Citado el: 26 de septiembre de 2012.] www.rae.es.

20. **Um, Taehoon , et al.** *A Quality Attributes Evaluation Method for an Agile Approach.* [pdf] Seoul, South Korea : Department of Computer Science & Engineering, Korea University, 2011.

21. **Swarup, M. Ben y Ramaiah, P. Seetha .** *An Approach To Modeling Software Safety.* [pdf] Visakhapatnam, India : Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008.

22. **Jeon, Sanghoon , y otros, y otros.** *Quality Attribute driven Agile Development.* [pdf] Korea : Ninth International Conference on Software Engineering Research, Management and Applications, 2011.

23. **Bass, Len, Clements, Paul y Kazman, Rick.** *Software Architecture in Practice. Second Edition.* [chm] s.l. : Addison Wesley, 2003. 0-321-15495-9.

24. **Vigil, Yamila Regalado.** *Metodología de evaluación de arquitectura de software.* [pdf] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2009.

25. **Billy Reynoso, Carlos.** *Introducción a la Arquitectura de Software.* Buenos Aires : s.n., Marzo de 2004.

26. **Cañete Valdeón, José Miguel .** *Desarrollo de la competencia "Pensamiento Analítico" mediante tácticas de arquitecturas software.* [pdf] Sevilla : Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla., 2010.

27. **Adolph, Steve, et al.** *Patterns for Effective Use Cases.* [pdf] s.l. : Cockburn-Highsmith Series Editor, 2001.

28. **Fowler, Martin.** *Patterns of Enterprise Application Architecture.* [chm] s.l. : Addison Wesley, 2002.

29. **Christopher Alexander, et al.** *A Pattern Language.* [doc] New York : Oxford University Press, 1977.

30. **Gamma, Erich, et al.** *Design Patterns: Elements of Reusable Object Oriented Software.* [pdf] s.l. : Addison Wesley, 1995.

31. **Bachmann, Felix .** *The principles of successful architecture evaluations.* [pdf] Pittsburg : Software Engineering Institute. Carnegie Mellon University, 2009.

32. **Bosch, Jan.** *Software Architecture Assessment.* [ppt] Netherlands : University of Groningen, 2001.

33. **Losavio, Francisca y Guillén-Drija, Christian .** *Comparación de métodos para la arquitectura del software: un marco de referencia para un método arquitectónico unificado.* [pdf] s.l. : Revista de la

Facultad de Ingeniería U.C.V, 2010. Vol.25. pp. 71–87.

34. **González, Aleksander, et al.** *Método de Evaluación de Arquitecturas de Software Basadas en Componentes (MECABIC).* [pdf] Caracas : Laboratorio de Investigación en Sistemas de Información (LISI). Universidad Simón Bolívar, 2005.

35. **Kazman, Rick, et al.** *SAAM: A Method for Analyzing the Properties of Software Architectures.* [pdf] Pittsburg : CMU/SEI, 1994.

36. **Kasman, Rick, Klein, Mark y Clements, Paul.** *ATAM: Method for architecture evaluation.* [pdf] Pittsburgh : Carnegie Mellon University. Software Engineering Institute., 2000. CMU/SEI-2000-TR-004.

37. **Clements, Paul C.** *Active Reviews for Intermediate Designs.* [pdf] Pittsburg : CMU/SEI, agosto de 2000. CMU/SEI-2000-TN-009.

38. **Kazman, Rick , Asundi, Jai y Klein, Mark .** *Quantifying the Costs and Benefits of Architectural Decisions.* [pdf] Pittsburgh : Software Engineering Institute. Dept. of Engineering and Public Policy.Carnegie Mellon University., 2001. PA 15213.

39. **Dolan, Thomas J.** *Architecture Assessment of Information-System Families a practical perspective.* [pdf] Eindhoven : Technische Universiteit Eindhoven, 2001. ISBN 90-386-0943-4.

40. **Lassing, Nicolaas Hendrik.** *Architecture-Level Modifiability Analysis. Tesis doctoral.* [pdf] Amsterdam : SIKS Dissertation. VRIJE UNIVERSITEIT, 2002. No. 2002-1..

41. **Williams, Lloyd G. y Smith, Connie U.** *PASA: A Method for the Performance Assessment of Software Architectures.* [pdf] Colorado : Performance Engineering Services and Software Engineering Research, 2002.

42. **Folmer, Eelke, Gurp, Jilles van y Bosch, Jan .** *Software Architecture Analysis of Usability.* [pdf] Holanda : University of Groningen, the Netherlands, 2003.

43. **Firesmith, Donald.** *QUality Assessment of System Architectures and their Requirements (QUASAR) Version 3.0.* [ppt] Pittsburgh : Carnegie Mellon University. Software Engineering Institute, 2008. PA 15213.

44. **Ambe, Mildred N. y Vizeacoumar, Frederick.** *Evaluation of two architectures using the Architecture Tradeoff Analysis Method (ATAM).* [pdf] 29 de abril de 2002.

45. **Babar, Muhammad Ali y Capilla, Rafael .** *Capturing and Using Quality Attributes Knowledge in Software Architecture Evaluation Process.* [pdf] 2008.

46. **González Alvarez, Larisa y Leyet Fernández, Osmar.** *Modelo de evaluación de arquitectura de software del proyecto ERP-Cuba.* [word] Ciudad Habana : Universidad de las Ciencias Informáticas (UCI), 2010. ISBN:978-959-286-011-7.

47. **Firesmith, Donald.** *QUASAR: A Method for the Quality Assessment of Software-Intensive System Architectures.* [pdf] Pittsburgh : Carnegie Mellon. Software Engineering Institute. , 2006. CMU/SEI-2006-HB-001.

48. **Leyet Fernández, Osmar.** *Propuesta metodológica para la obtención de los componentes de software en los proyectos del sistema Cedrux.* [pdf] Ciudad Habana : CEIGE. Universidad de las Ciencias Informáticas., 2011.

49. **Wagon, Stan.** *Mathematica in action: Problem-Solving Through Visualization and Computation.* 3ra. s.l. : Springer, 2010. ISBN: 0387753664.

50. **Ruskeepaa, Heikki .** *Mathematica navigator: Mathematics, Statistics, and Graphics.* 3ra. Finland : Academic Press, 2009. ISBN: 0123741645.

51. **Johnston, William y McAllister, Alex .** *A Transition to Advanced Mathematics: A Survey Course.* London : Oxford University Press, 2009. ISBN: 0195310764.

52. **Pressman, Roger.** *Software Engineering: A Practitioner's Approach.* 7th. Nem York : McGraw-Hill , 2010. ISBN 978-0-07-337 597 -7.

53. **Andrada Romero, Juan, et al.** *Estudios de uso e implantación sobre modelos de calidad de software. Norma ISO/IEC 9126.* [doc] Castilla la Mancha : Escuela Superior de Informática.Universidad de Castilla La Mancha., 17 de mayo de 2010.

54. **Project Management Institute, Inc.** *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* [pdf] Pennsylvania : PMI Publications, 2008.

55. **Leyet Fernández, Osmar y González Alvarez, Larisa.** *Modelo de producción para el flujo de arquitectura de sistema del proyecto ERP-Cuba.* [word] Ciudad Habana : Universidad de las Ciencias Informáticas (UCI), 2010. ISBN: 978-959-286-011-7.

56. **Firesmith, Donald.** *QUality Assessment of System Architectures and their Requirements (QUASAR).* [pdf] Pittsburgh, USA : Software Engineering Institute, Carnegie Mellon University, 18 de mayo de 2010. PA 15213.

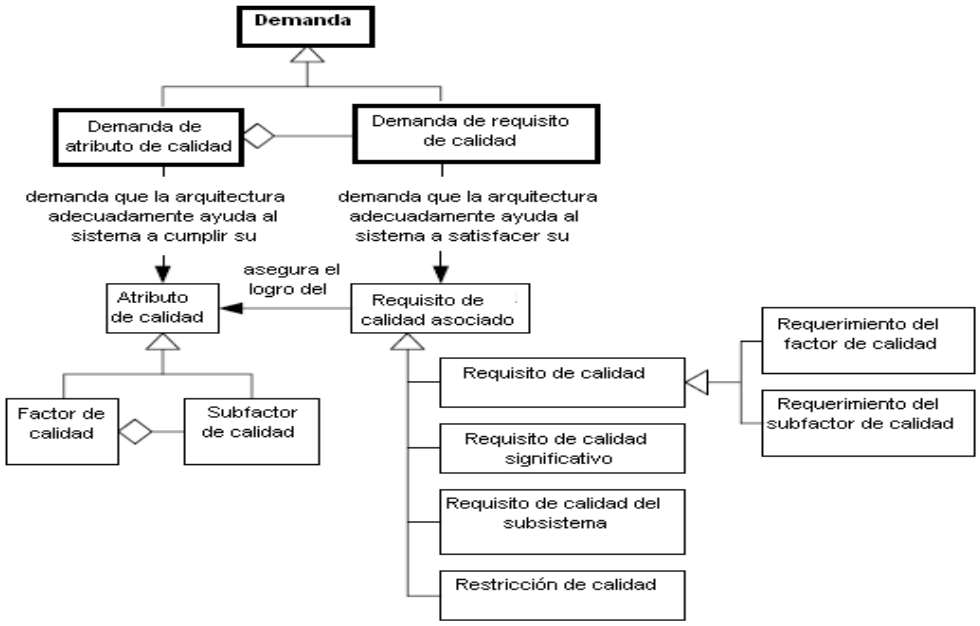
ANEXOS

Anexo 1. Fases y pasos del método de evaluación ATAM [6]

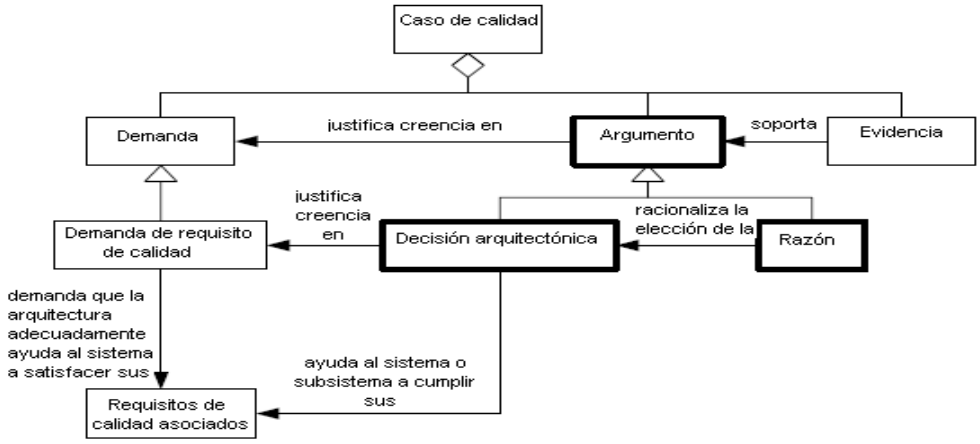
Fase 1: Presentación	
1. Presentación del ATAM	El líder de evaluación describe el método a los participantes, trata de establecer las expectativas y responde las preguntas propuestas.
2. Presentación de las metas del negocio	Se realiza la descripción de las metas del negocio que motivan el esfuerzo, y aclara que se persiguen objetivos de tipo arquitectónico.
3. Presentación de la arquitectura	El arquitecto describe la arquitectura, enfocándose en cómo ésta cumple con los objetivos del negocio.
Fase 2: Investigación y análisis	
4. Identificación de los enfoques arquitectónicos	Estos elementos son detectados, pero no analizados.
5. Generación del árbol de utilidad.	Se elicitan los atributos de calidad que engloban la "utilidad" del sistema (desempeño, disponibilidad, seguridad, modificabilidad, usabilidad, etc.), especificados en forma de escenarios. Se anotan los estímulos y respuestas, así como se establece la prioridad entre ellos.
6. Análisis de los enfoques arquitectónicos	Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos del paso 4. En este paso se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.
Fase 3: Pruebas	
7. Lluvia de ideas y establecimiento de prioridad de escenarios.	Con la colaboración de todos los involucrados, se complementa el conjunto de escenarios.
8. Análisis de los enfoques arquitectónicos	Este paso repite las actividades del paso 6, haciendo uso de los resultados del paso 7. Los escenarios son considerados como casos de prueba para confirmar el análisis realizado hasta el momento.

Fase 4: Reporte	
9. Presentación de los resultados	Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

Anexo 2. Composición de una demanda arquitectónica [47]



Anexo 3. Estructura de los argumentos arquitectónicos [47]



Anexo 4. Tipos de evidencia arquitectónica [47]