UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA DE SISTEMAS

Una Institución Adventista

Métodos y herramientas de trabajo para arquitecturas basado en componentes de desarrollo de software: Una revisión sistemática de la literatura.

Por:

Jeiner Manuel Castro Vásquez

Hans Alfredo Exebio Fernández

Asesor:

Mg. José Bustamante Romero

Lima, Diciembre del 2019

DECLARACIÓN JURADA
DE AUTORÍA DEL TRABAJO DE
INVESTIGACIÓN

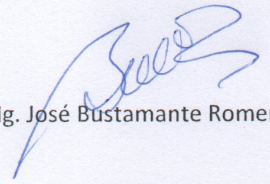
Mg. José Bustamante Romero, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas de la Universidad Peruana Unión.

DECLARO:

Que el presente trabajo de investigación titulado: "**MÉTODOS Y HERRAMIENTAS DE TRABAJO PARA ARQUITECTURAS BASADO EN COMPONENTES DE DESARROLLO DE SOFTWARE: UNA REVISIÓN SISTEMÁTICA DE LA LITERATURA**." Constituye la memoria que presentan los estudiantes Jeiner Manuel Castro Vásquez y Hans Alfredo Exebio Fernandez, para aspirar al grado de bachiller en Ingeniería de Sistemas, cuyo trabajo de investigación ha sido realizado en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este trabajo de investigación son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en Lima, el 04 Diciembre del 2019



Mg. José Bustamante Romero

Métodos y herramientas de trabajo para arquitecturas basado en componentes de desarrollo de software: Una revisión sistemática de la literatura

Trabajo de investigación

Presentado para optar al grado de bachiller en Ingeniería de Sistemas

JURADO CALIFICADOR

Mg. Sergio Omar Valladares Castillo
Presidente

Mg.Geraldine Verónica Alvizuri Llerena
Secretaria

Mg.Benjamin David Reyna Barreto
Vocal

Mg.Omar Leonel Loaiza Jara
Vocal

Mg. José Bustamante Romero
Asesor

Lima, 02 de diciembre del 2019

Métodos y herramientas de trabajo para arquitecturas basado en componentes de desarrollo de software: Una revisión sistemática de la literatura

Jeiner Manuel Castro Vásquez, Hans Alfredo Exebio Fernandez

Universidad Peruana Unión, UPeU NJ 08544, Perú
jeinercastro@upeu.edu.pe
hansexebio@upeu.edu.pe

Resumen: La arquitectura basada en componentes tiene un gran impacto en el desarrollo de software por motivos económicos y productividad laboral por la reutilización de componentes en diferentes proyectos de software, debido a esto se realizó una revisión sistemática de la literatura recopilando información mediante una estrategia rigurosa con el objetivo identificar métodos y herramientas de trabajo para una arquitectura de software basada en componentes. Primero se plantearon las preguntas de investigación en base a nuestro objetivo con la finalidad de encontrar los métodos y herramientas más actuales de esta arquitectura, luego se elaboró una matriz de términos y entidades para determinar las palabras clave de nuestra búsqueda, después se definió criterios de inclusión y exclusión a los artículos seleccionados para poder tener información más precisa. En cuanto a los resultados, del total de 576 artículos revisados coincidentes con los criterios de búsqueda y aplicando criterios de inclusión y exclusión, se identificaron 18 artículos que hacen referencia a 4 métodos que proponen un análisis de los componentes basado en un modelo, un ciclo de vida para una arquitectura basada en componentes por repositorios. También se encontraron 2 herramientas que apoyan el análisis por medio de diagramas y que agilizan el ensamblaje de los componentes a un proyecto de software. Se puede concluir que existen diversos métodos y herramientas que ayudan en la fase de análisis, construcción y pruebas para el desarrollo basado en componentes siendo el modelo arquitectónico Dedal y la herramienta DSL (Domain Specific Language) más usadas en el área de ingeniería de software basado en componentes.

Palabras claves: Arquitectura basada en componentes, DSL (Domain Specific Languaje), Modelo arquitectónico Dedal, Componentes arquitectónicos.

1 Introducción

En los últimos años, el desarrollo de software ha surgido como un mercado global para los negocios [1]. Así mismo también ha ido aportando mucho en la economía de los países y otras industrias como minería, medicina, medio ambiente, entre otros. Como resultado de la globalización, empresas de todo el mundo han comenzado a distribuir su entorno de desarrollo en diferentes sitios con el fin de producir mejores productos de software y más rápido sus servicios [2], de esta manera, poder automatizar procesos que cada organización tiene implementado.

Durante algunas décadas, los sistemas de software se han vuelto comparativamente más grandes y complejos que antes. Siguiendo el enfoque tradicional de esta nueva situación, surgieron problemas como el incumplimiento del plazo, el presupuesto y los requisitos de calidad [3]. Debido a esto, es muy importante que las empresas puedan automatizar los ciclos de vida de desarrollo de software para reducir el alto costo y la complejidad, de tal manera así poder brindar mayor confiabilidad a sus clientes. Por otro lado, el objetivo de la tecnología de desarrollo basado en componentes pretende construir sistemas de aplicación tomando elementos de una colección de componentes de software reutilizables (es decir, componentes fuera de la plataforma) y luego simplemente reunirlos [4]; de esta manera los ciclos de desarrollo de software serán más cortos, rentables y con un alto grado de calidad.

Para que un software cumpla los requisitos de calidad tanto en su funcionalidad, fiabilidad, eficiencia, usabilidad, etc es importante aplicar una arquitectura de software. La importancia de la arquitectura de software es que no sólo define el sistema en términos de componentes, relaciones y reglas o condiciones, sino que también proporciona una estructura para el software y gestiona el tamaño y la complejidad de los grandes sistemas de software [5]**.** Como resultado se podrá disminuir considerablemente los riesgos que puedan presentarse en las organizaciones. Sin embargo, la mayoría de los errores de diseño inducidos durante el desarrollo de la arquitectura de software solo son visibles en las etapas posteriores del desarrollo [6]. Esto conlleva a los altos costos producidos por la reelaboración.

La ingeniería de software basada en componentes (CBSE) es un procedimiento de incorporación del marco de software diferente para estructurar una aplicación con el fin de cumplir con la utilidad [7]. CBSE tiene como objetivo crear software a partir de componentes preexistentes, crear componentes como entidades reutilizables y evolucionar aplicaciones mediante la sustitución de componentes [8]. este enfoque requiere cambios significativos en el entorno de desarrollo incluso desde el punto de vista técnico como empresarial. Las motivaciones detrás de CBSE han sido de carácter

empresarial y técnico, es decir, una mayor eficiencia y eficacia, menores costos y un menor tiempo de comercialización, por un lado, y una mejor calidad con respecto a menos errores, mejor rendimiento, mantenibilidad, portabilidad, etc [9].

Un componente de software se divide en dos categorías: componentes atómicos y componentes compuestos [10]. Es decir, un componente atómico es una pieza de software prefabricada que no necesita subdividirse en componentes más pequeños; mientras que un componente compuesto está formado por un grupo de componentes relacionados. Las razones principales para la producción e implementación de componentes de software son: separabilidad de los componentes de su contexto; desarrollo independiente de componentes, pruebas, configuración y posterior reutilización; actualización y reemplazo en sistemas en ejecución [11].

La revisión sistemática de la literatura surge a partir de la necesidad de querer identificar qué métodos y herramientas de trabajo para arquitecturas de software basada en componentes existen para luego poder incluirlas en los proyectos de desarrollo de software y qué ventajas ofrecen. Esta necesidad se fundamenta en la creciente complejidad, evolución y diversidad de métodos y herramientas de soporte disponible para el desarrollo de software, lo que supone un reto para la industria del software al momento de trabajar con esta arquitectura.

Este artículo está distribuido de la siguiente manera: la sección 2 describe la revisión sistemática de la literatura, la sección 3 presenta los métodos de la revisión sistemática de la literatura, la sección 4 describe los resultados de la revisión y finalmente la sección 5 describe las conclusiones y trabajo futuro.

2 Revisión de la literatura

2.1 Arquitectura de software

Perry y Wolf son los primeros en proponer un modelo de arquitectura de software como una forma de oponerse a la nueva complejidad de los sistemas (estructuras de datos) que consiste en tres componentes: elementos, forma y razón [12]. Este trabajo es el punto de partida para lo que hoy conocemos como arquitectura de software. Así mismo, El diseño de software recibió una gran atención por parte de los investigadores en la década de 1970 [13]. esta atención surgió como respuesta a los problemas complejos de desarrollo de software.

En la actualidad la arquitectura de software es la base primordial para un buen sistema de software. En el corazón de todo sistema de software bien diseñado se encuentra una buena arquitectura de software, un buen conjunto de decisiones de diseño [14]. Es decir, es el principal enfoque de la ingeniería de software puesto que la producción de productos de software de alta calidad depende de las decisiones principales. Los arquitectos de software son diseñadores que tienen una visión de alto nivel tanto en aspectos comerciales como técnicos y, entre otras cosas, trabajan con muchas partes interesadas [15]. Toman decisiones sobre qué estilo de arquitectura usar, como diseñar una API o qué métodos deben incluirse en una clase, etc.

La toma de decisiones (DM) en arquitectura de software es un tema que rara vez se estudia en ingeniería de software. Los aspectos humanos de la DM es un término general que describe lo que sucede cuando los humanos toman decisiones en el diseño de software [16]. Estas decisiones pueden estar influenciados por diferentes factores como la cognición, los prejuicios, el pensamiento grupal, la comunicación. Se espera que la arquitectura garantice la satisfacción de los requisitos, sin una consideración especial de los cambios en el contexto que pueden violar los supuestos con respecto a la inmutabilidad de los requisitos [17].

2.1.1 Desarrollo basado en arquitectura

Los estilos y patrones arquitectónicos definen la forma de organizar los componentes del sistema para que uno pueda construir un sistema completo y cumplir con los requisitos del cliente [18]. MVC describe una aplicación CUI en términos de tres abstracciones fundamentales: modelos, vistas y controladores. Sin embargo, wing MVC es una versión especializada del MVC clásico, destinado a admitir un aspecto conectable en lugar de aplicaciones en general. La arquitectura de componentes Swing puede considerarse transparente, lo que significa que uno puede usar un componente sin tener que preocuparse por su estructura interna [19]. Por otra parte, La Arquitectura basada en microservicios es una de las arquitecturas de software más populares, puesto que; una arquitectura de microservicio permite que un servicio se divida en un número de tamaño más pequeño pero que se ejecuta simultáneamente [19]. A diferencias de una arquitectura orientada a servicios (SOA) que se implementa frecuentemente en aplicaciones monolíticos es decir que se implementa en una sola unidad.

Hay varios estilos y patrones arquitectónicos disponibles en la industria del software. por lo que uno necesita entender qué estilo de arquitectura será apropiado para su proyecto. En la tabla 1 se observa una categorización completa de todos los estilos arquitectónicos existentes:

Tabla 1 Descripción de los estilos arquitectónicos más importantes

Tipo de aplicación	Estilo arquitectónico
Memoria compartida	1.- Blackboard 2.- Data-centric 3.- Rule-based
Sistema distribuido	1.- Client-server 2.- Space based architecture 3. -Peer-to-peer 4.- Shared nothing architecture 5.- Broker 6.- Representational state transfer 7.- Service-oriented
Mensajería	1.-Event-driven 2.-Asynchronous messaging 3.-Publish-subscribe
Estructura	1.- Component-based 2.- Pipes and filters 3.- Monolithic application based 4.- Layered
Adaptable System	1.- Plug-ins 2.- Reflection 3.- Microkernel
Sistema moderno	1.- Architecture for Grid Computing 2.- Multi-tenancy Architecture 3.- Architecture for Big-Data

2.2 **Desarrollo de software basado en componentes**

El desarrollo de software basado en componentes es una alternativa de diseño que favorece la reutilización de elementos software y facilita el desarrollo de sistemas a partir de elementos preexistentes [20]. Estos componentes surgieron de la necesidad de descomponer requerimientos para la reutilización próxima un software diferente [21]. Es decir, que el proceso de reutilización de componentes es la pieza clave de una ingeniería de software basado en componentes (CBSE).

CBSD propone nuevo enfoque en el diseño, construcción, implementación. Sobre todo, porque se ensamblan de diferentes repositorios, además, se ejecutan en diferentes lenguajes y en varias plataformas [22]. Este nuevo enfoque de CBSD existen diferentes modelos que están ligados a una implementación en específico. Un modelo de componente de software es una definición de 1) semántica de los componentes, es decir, qué componentes deben ser ensamblados, de acuerdo a las necesidades del cliente; 2) la sintaxis de los componentes, es decir, cómo son definidos, construidos y representados y 3) la composición de los componentes, es decir, cómo es compuesto o ensamblado [23].

En la figura 1 se puede observar un ejemplo sobre componentes reutilizables. Se muestra dos sistemas totalmente diferentes, un sistema de ventas y un sistema académico, cada sistema posee requerimientos diferentes, así mismo, cada componente es diferente, no obstante, pueden estar compuesto por varios componentes o ser relacionados de manera independiente. Por ejemplo, el componente “Inicio de sesión de usuario” y el componente “Información de perfil de usuario” son requisitos que la mayoría de sistemas necesitan, por lo que puede ser integrado en cualquier sistema.

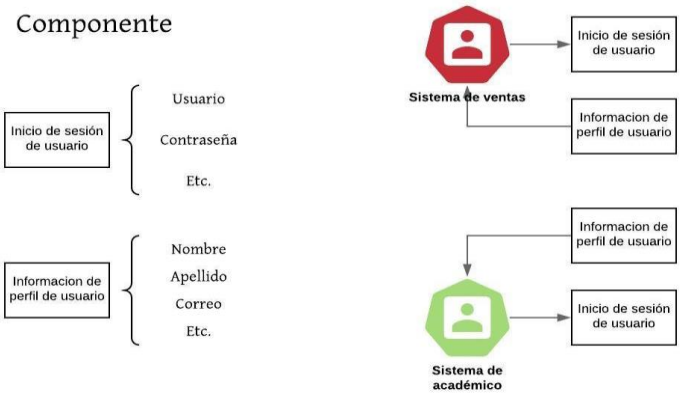


Figura 1. componentes integrados en diferentes sistemas

2.3 Componente

2.3.1 Análisis de requisitos de componentes

La ingeniería de requisitos (RE) es la fase inicial y crítica de la ingeniería de software. El éxito del proyecto de software se basa en buenas prácticas de ingeniería de requisitos [24]. el propósito de análisis de requisitos es tener todas las necesidades de los usuarios y documentarlas para mayor claridad y para futuras referencias; así mismo también determinar restricciones, limitaciones de los usuarios para el sistema.

El análisis de requisitos de componentes es el procedimiento para encontrar, comprender, archivar, aprobar y tratar las necesidades de un segmento [7], es decir, crear prerrequisitos completos que sean predecibles para el sistema y significativos para los interesados del proyecto. Y además el dialecto de programación la etapa y las interfaces identificadas con el componente.

Un componente se define como un conjunto de clases relacionadas que proporciona una funcionalidad relativamente gruesa y realiza interfaces que especifican firmas y restricciones [25]. En la figura 2 se muestra elementos de un componente y su metamodelo en UML. Un componente agrega un flujo de trabajo es decir se hace un análisis de cómo se realizan las tareas, cual es el orden de ejecución, como fluye la información que soporta las tareas(métodos, clases, etc.); así mismo un componente es dividido en dos tipos 1) componente de caja blanca, es decir que se conoce su funcionalidad interna 2) Componentes de caja negra lo cual no se conoce su funcionalidad interna y no puede ser modificado. Seguidamente un componente de software está relacionado con una interfaz, también cumple una funcionalidad u operaciones. Una funcionalidad está conformada por un conjunto de clases.

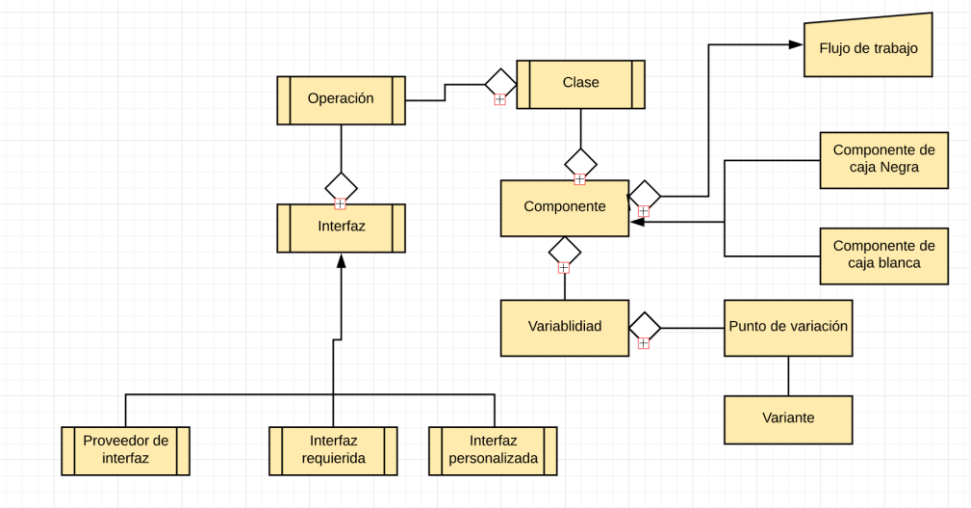


Figura 2. Metamodelo de un componente

2.3.3 Atributos de un componente

Un componente debe cumplir con ciertos atributos para que posteriormente pueda ser utilizado con facilidad en el proceso de desarrollo de software logrando una gestión eficaz en el desarrollo del sistema. Según Deepti Negi [26] menciona 17 atributos que debe tener un componente. En la tabla 2 se puede observar los atributos de un componente.

Tabla 2 Atributos de un componente de software

Nª	Atributos	Descripción
1	**Funcionalidad**	Es la capacidad que tiene un componente para que pueda cumplir con éxito los requisitos deseados.
2	**Ejecución correcta**	Después de que un componente se haya ejecutado exitosamente, se debe asegurar que el programa completado realizó la función requerida para realizar una tarea deseada.
3	**Usabilidad**	La capacidad de aprender, comprender y operar un componente fácilmente para obtener el resultado deseado por usuario final.
4	**Identidad**	Para que un componente sea identificable, utilizamos una convención de nomenclatura en cada fase del ciclo de vida del desarrollo de software, es decir, en las fases de desarrollo, prueba, reutilización despliegue, operación y mantenimiento.
5	**Modularidad**	Es el grado en el cual los componentes de un sistema pueden estar separados, es decir un desarrollador debe enfocarse en en construir unidades distintas individuales llamadas módulos.
6	**Encapsulación**	Capacidad de permitir ocultar la información interna del mundo exterior, es decir, todos los métodos y datos requeridos dentro de una sola unidad se conoce como encapsulación.
7	**Interface**	La interfaz de un componente es un medio de interconexión con otros componentes seleccionados para su comunicación.
8	**Independiente**	Separación de todas las responsabilidades de los antecedentes operativos con el propósito de integración y despliegue
9	**Reusabilidad**	Los componentes pueden ser fácilmente reutilizados por diferentes clientes en sus diferentes entornos de aplicación
10	**Portabilidad**	Es la capacidad de un componente para ejecutarse con una modificación mínima en diferentes plataformas de hardware o software.
11	**Documentación**	Una documentación bien definida consiste en especificaciones sobre el software.
12	**Personaliza-Bilidad**	Es la capacidad de modificar un componente de software existente o una parte de él para cumplir con los requisitos del cliente. La personalización a implicar cambios de configuración en diversos aspectos del componente (es decir, funciones de componentes, interfaces etc.) Y empaquetado para nuevas implementaciones
13	**Capacidad de despliegue**	Después de un desarrollo exitoso, el componente se implementa para hacerlo funcional. La implementación del software implica empaquetar, instalar y hacer que los componentes estén listos para ejecutarse en un nuevo entorno.
14	**Interoperabilidad**	Es la capacidad de un componente para comunicarse e intercambiar

		información con otros componentes de diferentes procesos localmente o a través de la red. La interoperabilidad es una característica importante del diseño de componentes.
15	**Capacidad de integración**	Esta característica del diseño de un componente facilita la combinación de varios componentes para formar un sistema complejo. Se centra en la personalización, composición y configuración de componentes para combinar y obtener un sistema de software completo.
16	**Probabilidad**	Esta característica del diseño de un componente facilita la combinación de varios componentes para formar un sistema complejo. Se centra en la personalización, composición y configuración de componentes para combinar y obtener un sistema de software completo.
17	**Estándares estandarizados**	Para garantizar la aceptación generalizada de los componentes, debemos centrarnos en su estandarización para obtener estándares y modelos uniformes para el desarrollo, las pruebas, el control de calidad, la reutilización, la implementación, la gestión de proyectos, etc.

En la figura 3 se observa un bosquejo sobre cómo se realiza un proceso de ensamblaje de software a nivel general, es decir, para la construcción de un software necesitamos de un repositorio de componentes para ello se realiza un proceso de selección de componentes, esta selección va a depender de las necesidades del proyecto. Así mismo, de ser necesario se podrá adquirir componentes de terceros (Componentes comerciales COTS) para cubrir algunas necesidades complejas del proyecto.

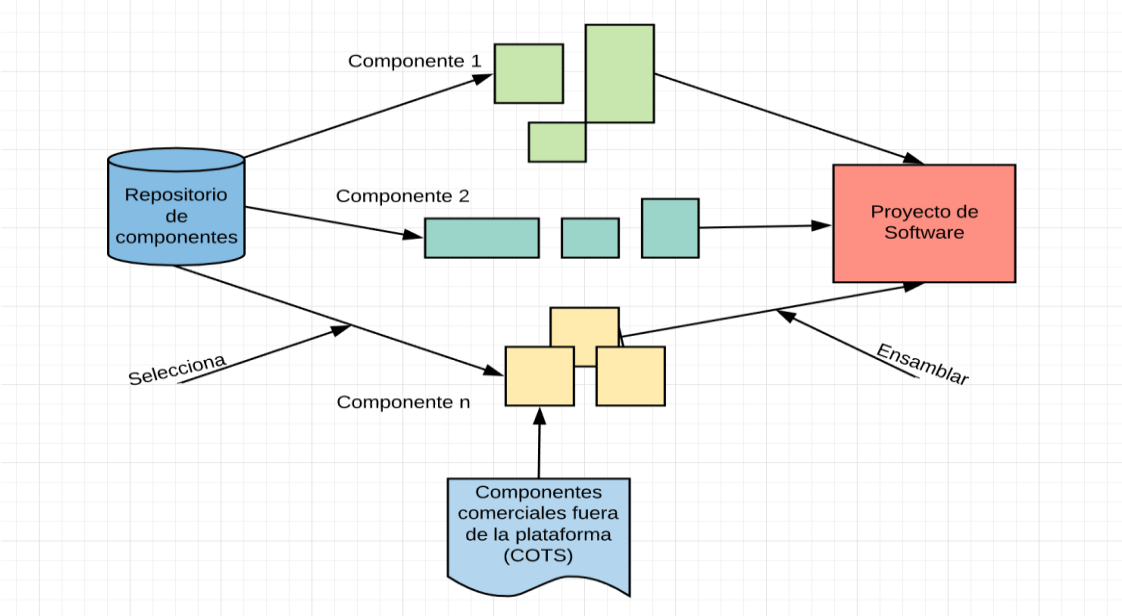


Figura 3. Estructura fundamental de la CBSE

Emplear un modelo para el ciclo de vida de desarrollo de software basado en componentes conlleva a que los gerentes del proyecto puedan optimizar el proceso de construcción del ciclo de vida de desarrollo de software. Un modelo de objeto para componentes se clasifica en cuatro aspectos: granularidad, visibilidad, persistencia de datos y sincronización de métodos [21]. Primero la granularidad de los componentes se clasifica en objetos individuales y objetos compuestos por número de objetos. Segundo la usabilidad de los componentes se clasifica en caja blanca y caja negra según si se puede acceder directamente al interior de los componentes.

En la figura 4 se muestra un sistema basado en servicios web, sin embargo, solo funciona para aquel sistema, siendo incapaz de poder emplearse en otros sistemas con una visión de negocio diferente. los sistemas restantes están desarrollados en base a componentes, esto permite que se pueda reutilizar para otros proyectos.

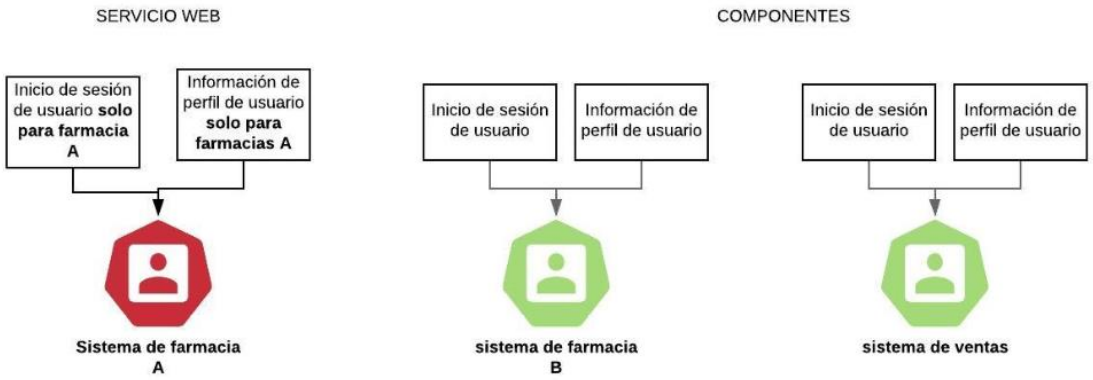


Figura 4. Estructura fundamental de la CBSE

3 Método de la revisión sistemática de la literatura

Revisión sistemática de la literatura es una metodología enfocada en recolectar información de un objeto de estudio específico y analizar así mismo esta metodología reúne un conjunto de procedimientos definidos en un protocolo de revisión que especifica los pasos a seguir.

3.1 Necesidad de la revisión sistemática

La revisión sistemática de la literatura surge a partir de la necesidad de querer identificar qué métodos y herramientas de trabajo para arquitecturas de software basada en componentes existen para luego dichos métodos y herramientas puedan ser aplicados en los diferentes fases como requisitos, ensamblaje, pruebas de componentes de dicha arquitectura y posteriormente los equipos de desarrollo puedan aplicar en proyectos de software basados en componentes reales con la finalidad de reducir tiempos y costos en la producción de software, así mismo identificar qué ventajas ofrecen. Esta necesidad se justifica con el excesivo tiempo de desarrollo, y el alto costo de los proyectos de software, evolución y diversidad de métodos y herramientas de soporte disponible para el desarrollo de software, lo que supone un reto para la industria del software al momento de trabajar con esta arquitectura.

Además, se requiere identificar, las características que son más relevantes y determinantes al momento de decidir qué métodos y herramientas de trabajos para arquitectura de software basado en componentes son los más adecuados o se ajustan más a un determinado proyecto; y que atributos formarían parte de los criterios bajo las cuales se evaluarán dichas herramientas.

De acuerdo a la plantilla Goal, Question, Metric (GQM) para establecer el objetivo de la investigación se pueden observar los siguientes componentes en la tabla 3.

Tabla 3 Elaboración del objetivo de la investigación

Campo	Valor
Objeto de estudio	Arquitectura de componentes
Propósito	Identificar
Foco	Métodos, técnicas, prácticas, metodologías, procesos y herramientas
Involucrados	Desarrollo de software basado en componentes, desarrollo de software, pequeñas y medianas empresas de desarrollo de software, ingeniería de software basado en componentes, arquitectura de software, servidores, Arquitectura de software basado en componentes, framework de desarrollo
Factores de contexto	Ninguno para este caso

3.2 **Preguntas para la revisión sistemática**

Para la elaborar de las preguntas de investigación se tomó como referencia el objetivo de la investigación que se encuentra en la tabla anterior. En la tabla 4 se muestra las preguntas propuestas y la motivación. Adicionalmente en la tabla 5 se observa las preguntas de bibliometría propuestas con la finalidad de obtener visibilidad sobre la evolución y tendencia de los estudios:

Tabla 4 Preguntas de investigación y motivación

Id	Pregunta	Motivación
PI-1	¿Qué métodos de trabajo son adecuados para una arquitectura de software basada en componentes?	Identificar qué métodos de trabajo son adecuados para la construcción de una arquitectura basada en componentes.
PI-2	¿Qué herramientas de modelado de software, construcción e implementación existen para una arquitectura de software basada en componentes?	Determinar las herramientas adecuadas para una arquitectura de software basada en componentes

Tabla 5 Preguntas de bibliometría

Id	Pregunta	Motivación
PB-1	¿Cuál es la cantidad de publicaciones sobre el tema?	Determinar la cantidad de estudios publicados sobre este tema
PB-2	¿En qué medida ha evolucionado las publicaciones sobre este tema?	Identificar la frecuencia de publicaciones para poder establecer la relevancia del tema en el tiempo
PB-3	¿Cuáles son las publicaciones en las que se ha encontrado estudios relacionados al tema?	Identificar en qué dominio de aplicación se concentra la mayor cantidad de publicaciones sobre este tema

3.3 **Definición de las cadenas de búsqueda**

Para la elaboración de la cadena de búsqueda se eligió la estrategia PICO en el cual se realizaron los ajustes necesarios para la selección de resultados. En la tabla 6 se presenta la estrategia PICO.

Tabla 6 Estrategia cadena de búsqueda PICO

Población	Intervención	Resultados
Entidad: Métodos y herramientas de trabajo para arquitectura de software basado en componentes **Término principal 1**: Métodos. **Términos alternos:** Metodologías, procedimientos, normas, buenas prácticas, reglas, modelos, marco de trabajo, diseño de trabajo. **Justificación**: Se selecciona el término por ser el objeto de estudio de la revisión a ejecutar y se obtienen los términos alternos que representan las variantes o cercanos al término principal. **Término principal 2**: Herramientas **Términos alternos**: Instrumentos, mecanismos y artefactos. **Justificación**: Se selecciona el término por ser el tipo de análisis a ejecutar y se obtienen dichos términos alternos por ser aquellos los diferentes tipos de comparaciones existentes.	**Entidad**: Aplicado a arquitecturas de software basado en componentes **Término principal 1**: Arquitecturas basado en componentes **Términos alternos**: Ciclo de desarrollo de software, Arquitectura de software **Justificación**: Se selecciona el término por ser el elemento sobre el cual se realizará la revisión sistemática y se obtienen dichos términos alternos por ser aquellos los tipos de objetos. **Término principal 2:** Componentes de desarrollo **Términos Alternos:** Servicio web, desarrollo web, Microservicios **Justificación**: Se selecciona el término por ser el elemento sobre el cual se realizará la revisión sistemática y se obtienen dichos términos alternos por ser aquellos los tipos de objetos.	**Entidad**: Propuestas, experiencias, sobre métodos y herramientas de trabajo para desarrollo de una arquitectura basado en componentes, Modelos **Término principal:** Experiencias **Términos alternos**: Propuestas, aplicaciones, implementaciones **Justificación**: se seleccionan dichos términos puesto que es lo que se busca obtener como resultado de la investigación.

Idioma: Se eligió el inglés y español como idioma para la cadena de búsqueda puesto que es de los idiomas más utilizados para la elaboración de artículos.

Siguiendo las fases de la estrategia pico se obtuvo como resultado la cadena de búsqueda a partir del uso de operadores booleanos: (Población) AND (Intervención) AND (Comparación) AND (Resultado). En la tabla 7 se puede apreciar los elementos de la estrategia PICO a partir de los cuales se elabora la cadena de búsqueda.

Tabla 7 Términos en inglés y español con conectores lógicos a ser usados en la búsqueda

Concepto	Términos en inglés
Población	(Method * or * methodology * or * process) and (tools * or * mechanism) or (Método* or * metodología * or * proceso) and (herramientas* or * mecanismo)
Intervención	(“component based architecture" * or * "service web”) and (“component-based software engineering”) or (“arquitectura basada en componentes” * or * “servicio web”) and (“ingeniería de software basada en componentes”)
Comparación	
Resultado	(propos* or * experi) or (proponer * or * experiencia)

Tipos de búsqueda: Se realizó una búsqueda en las librerías digitales previamente seleccionadas de acuerdo a su relevancia en el ámbito científico y al contexto que se requiere evaluar.

3.4 Criterios de inclusión y exclusión

Después de ejecutar la cadena de búsqueda en las diferentes bases de datos. los resultados fueron sometidos a evaluación para poder identificar cuáles son los estudios primarios que responden directamente a las preguntas de investigación formulado tomando como en consideración los criterios de inclusión y exclusión como se muestra en la tabla 8.

Tabla 8 Criterios de inclusión y exclusión

Criterios de inclusión	*Criterios de Exclusión*
CI 1. Se consideran todos aquellos artículos provenientes de librerías digitales indexadas.	CE 1. Serán excluidos los artículos duplicados.
CI 2 los artículos deben provenir del área de Ingeniería de Software.	CE 2 Serán rechazados los artículos que no se encuentren en idioma inglés y español.
CI 3. Se considerarán todos los artículos que se encuentren dentro del rango de temporalidad definido (No mayor de 10 años).	CE 3 Serán rechazados los artículos de contenido similar, quedándose solo los que tengan el contenido más completo.
CI 4. Se aceptarán artículos provenientes de revistas científicas y conferencias.	CE 4 Serán excluidos los estudios secundarios, estudios terciarios y resúmenes.

Temporalidad: Se consideración los estudios desarrollados en los últimos 15 años debido que se requiere analizar métodos y herramientas de software que se mantengan vigentes.

Fuentes datos: Las librerías digitales indexadas consideradas por su relevancia científica para la selección de artículos fueron:

- IEEE Xplore (https://ieeexplore.ieee.org/Xplore/home.jsp)
- ScienceDirect (https://www.sciencedirect.com/)
- SCOPUS (https://www.scopus.com/home.uri)

Procedimientos para la selección de estudios: Para la selección de los artículos RSL se sigue las siguientes fases:

- Fase 1: Se procede a la ejecución de la cadena de búsqueda en las bases de datos indexadas que fueron seleccionados aplicando los criterios de inclusión y exclusión en la tabla 9.
- Fase 2: Se realizó la revisión de títulos de los artículos de la ejecución del paso 1 excluyendo los que no fueron totalmente relevantes con el objeto de estudio.

- Paso 3: Se realizó la revisión de los resúmenes de los artículos que fueron seleccionados en el paso 2 para luego proceder con la exclusión según los criterios definidos en la tabla 9.
- Paso 4: Finalmente se realizó una revisión preliminar del contenido de los artículos que fueron seleccionados en el paso anterior para luego aplicar los criterios de selección según la tabla 9.

Tabla 9 Procedimiento y criterios de inclusión y exclusión

Procedimiento	Criterio de selección
Paso 1	CI 1, CI 4, CE 3, CI 3, CE 1, CE 2, CI 2
Paso 2	CI 1, CI 2, CE4 CE 5
Paso 3	CE3, CI4
Paso 4	CE3, CE4

3.5 Criterios de calidad

Se procedió con la definición del esquema de evaluación de calidad, en el cual se evaluó la calidad de los estudios seleccionados. Dentro del esquema se estableció una lista de criterios con la finalidad de comprobar el cumplimiento de cada artículo. Cada criterio que fue establecido está acompañado con la escala de evaluación de Rouhani que consiste en los siguientes puntajes: Si cumple (S) = 1, Cumple parcialmente (P) = 0.5 y no cumple (N) = 0.

Se realizó una estrategia para la extracción de datos a través del diseño de un formulario, seguidamente el método que se usó para esta RSL consiste en la realización de una síntesis narrativa basada en el marco propuesto por Popay, lo cual seguirá los siguientes pasos como se muestra en la tabla 10.

Tabla 10 Criterio de evaluación de calidad

Nª	Criterio de evaluación de calidad
1	¿El método seleccionado para llevar a cabo el estudio ha sido documentado adecuadamente? S: El método seleccionado ha sido documentado apropiadamente. P: El método seleccionado ha sido documentado parcialmente. N: No se ha documentado el método seleccionado.
2	¿El estudio aborda las amenazas a la validez? S: El estudio aborda las amenazas totalmente. P: El estudio aborda las amenazas parcialmente. N: No se detallan amenazas.
3	¿Se han documentado las limitaciones del estudio de manera clara? S: Las limitaciones se han documentado claramente. P: Las limitaciones se han documentado parcialmente. N: No se han documentado las limitaciones.
4	¿Los aportes del estudio para las comunidades científica, académica o para la industria han sido descritos? S: Los aportes del estudio han sido mencionados claramente. P: Los aportes del estudio han sido mencionados parcialmente. N: No se han mencionado aportes.
5	¿Los resultados han contribuido a responder las preguntas de investigación planteadas? S: Los resultados han contribuido a responder todas las preguntas de investigación. P: Los resultados han contribuido a responder algunas preguntas de investigación. N: Los resultados no han contribuido a responder las preguntas de investigación.

Validar el protocolo de investigación. El protocolo utilizado para el desarrollo de la RSL fue revisado en primer lugar por el asesor designado y posteriormente los dictaminadores asignados.

4 Resultados

En esta sección se procede con la descripción a detalle de todos los pasos ejecutados.

4.1 Resultados de la búsqueda

En la tabla 11 se muestran los resultados y las búsquedas que fueron aplicados. Se realizaron algunos ajustes de acuerdo al siguiente detalle.

- La base de datos IEEE Xplore filtraba muchos datos no relevantes por lo que se optó en eliminar parte de los caracteres de la cadena para afinar la búsqueda.
- De igual manera se eliminó parte de los caracteres para así afinar la búsqueda en la base de datos SCOPUS por el motivo de que filtraba muchos datos no relevantes

Tabla 11 Resultado de la búsqueda

Base de Datos	Fecha	Total
Cadena de Búsqueda		
IEEE XPLORE	Junio 2019	13
TITLE (Method * or * methodology * or * process) and (tools * or * mechanism) or (Método* or * metodología * or * proceso) and (herramientas* or * mecanismo) and (“component based architecture" * or * "service web”) and (“component-based software engineering”) or (“arquitectura basada en componentes” * or * “servicio web”) and (“ingeniería de software basada en componentes”) and (propos* or experi*) or (proponer * or * experiencia)		
SCOPUS	Junio 2019	0
TITLE (Method * or * methodology * or * process) and (tools * or * mechanism) or (Método* or * metodología * or * proceso) and (herramientas* or * mecanismo) and (“component based architecture" * or * "service web”) and (“component-based software engineering”) or (“arquitectura basada en componentes” * or * “servicio web”) and (“ingeniería de software basada en componentes”) and (propos* or experi*) or (proponer * or * experiencia)		
SCIENCEDIRECT	Junio 2019	5
TITLE (Method * or * methodology * or * process) and (tools * or * mechanism) or (Método* or * metodología * or * proceso) and (herramientas* or * mecanismo) and (“component based architecture" * or * "service web”) and (“component-based software engineering”) or (“arquitectura basada en componentes” * or * “servicio web”) and (“ingeniería de software basada en componentes”) and (propos* or experi*) or (proponer * or * experiencia)		

4.2 **Resultados de filtros aplicados**

Los artículos encontrados en todas las bases de datos fueron exportados en formato pdf, a continuación, se presenta el detalle de la serie de pasos ejecutados en la selección del estudio.

Paso 1: La lista de artículos resultado de la búsqueda fue ordenada por el título con la finalidad de eliminar todos aquellos artículos duplicados. La base de datos que produjo la mayor cantidad de resultados fue IEEE XPLORE.

Paso 2: En cuanto a la lista de resultados provenientes en el paso 1, se revisaron los títulos de cada artículo y en base a eso proceder con la exclusión de los artículos no relevantes de acuerdo a lo establecido en los criterios según la tabla V. Los títulos que quedaron en duda se consideraron en la lista con un alias para poder identificarlo en el siguiente paso.

Paso 3: Los artículos provenientes del paso 2 fueron revisados principalmente el resumen y excluidos de acuerdo a lo establecidos en la tabla 12.

Paso 4: Para proceder con la revisión del contenido de los artículos provenientes del paso 3, se procedió con la descarga de los artículos completos de la base de datos SCOPUS, SCIENCEDIRECT y IEEE XPLORE luego se procedió con la revisión preliminar de los artículos descargados. Para luego proceder con la exclusión de los artículos que no tenían relevancia alguno de acuerdo a los criterios establecidos en la tabla 12.

En la tabla 12 se muestran los resultados de la selección de estudios detallando todos los artículos restantes de dicha selección.

Tabla 12 Resultado de la búsqueda

BASE DE DATOS	ARTÍCULOS DESCUBIERTOS	PASO 1	PASO 2	PASO 3	PASO 4
IEEE XPLORE	342	156	56	24	13
SCIENCEDIRECT	164	50	25	5	5
SCOPUS	1	0	0	0	0
Total	507	0	0	0	18

Sobre un total de 18 artículos resultantes se aplicó la lista de criterios establecidos en la sección 3. Los resultados de la evaluación se muestran en la tabla 13.

Tabla 13 Extracción de datos de un estudio primario

Criterio	Detalle	Relevancia
Identificador	1	-
Fuente	IEEExplore	PB-I
Título	A Study of Component Based Software System Metrics	PI-2
Autores	Sakshi Patel, Jagdeep Kaur	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc	PBI-3
Año de publicación	2017	PBI-2
Tipo de publicación	International Conference	PBI-2
Objetivo de análisis	Estudiar métricas basada en componentes para proporcionar reutilización y disminuir el costo y el tiempo de desarrollo.	PI-I

4.3 **Análisis bibliométrico**

En esta parte se describe el análisis sobre los artículos seleccionados para esta RSL de acuerdo a los factores como tiempo, tipo de artículo y tema tratado, así mismo como la respuesta para cada una de las preguntas de investigación o bibliométricas

1. Pregunta de bibliometría 1 (PB-1)

   ¿Cuál es la cantidad de publicaciones sobre el tema?

En la figura 5 se muestra el porcentaje en cuanto a la cantidad de publicaciones según las librerías digitales seleccionadas que son IEEE Xplore, ScienceDirect y SCOPUS. Podemos observar que la librería de IEEE Xplore representa un 68% del total de los artículos relacionados al objeto de estudio, a diferencias de otras librerías se pudo notar que IEEE Xplore contiene mayor información referente al objeto de estudio, gran cantidad de información se encuentra en idioma inglés, Seguidamente tenemos a la librería de ScienceDirect con un 32% del total de los artículos relacionados al tema, finalmente Scopus con 0% de artículos relacionados al tema. En base a este análisis se puede inferir que las conferencias, revistas científicas y artículos de investigación son la mayor fuente de los estudios referente al objeto de estudio.

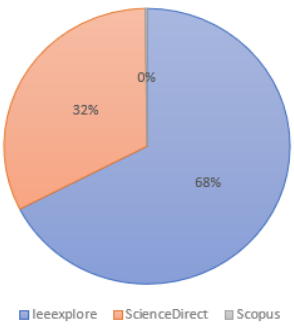


Figura 5. Cantidad de publicaciones sobre el tema

2. Pregunta de bibliometría 2 (PB-2)

¿En qué medida ha evolucionado las publicaciones sobre este tema?

Los resultados obtenidos en cuanto a la ejecución de la cadena de búsqueda y la selección mostrados en la tabla 12 se puede observar en la figura 6 el incremento en el número de publicaciones sobre los métodos y herramientas para arquitecturas basada en componentes de desarrollo de software a partir del año 2008 en adelante hasta el 2010, porque las publicaciones de los artículos se centraron en modelos y herramientas para aplicaciones de escritorio como por ejemplo Java Beans [25]. Durante los años 2011 y 2013 hubo pocas publicaciones sobre métodos o herramientas relacionado con el desarrollo de componentes. Sin embargo, durante los años de 2009 al 2015 junto con los avances de los servicios web y la computación en la nube [27] , las publicaciones incrementaron aunque no tanto como en el año 2008. En el año 2018 solo se encontró un artículo relacionado al tema.

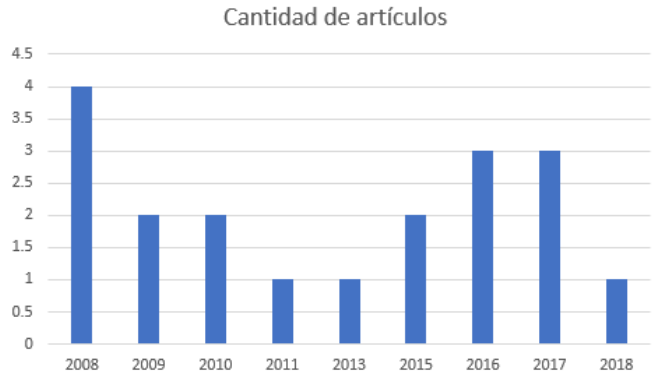


Figura 6. Evolución de las publicaciones sobre Métodos y herramientas de trabajo para arquitecturas basado en componentes de desarrollo de software

3. Pregunta de Bibliometría 3(PB-3)

¿Cuáles son las publicaciones en las que se ha encontrado estudios relacionados al tema?

En la tabla 14 se presentan el título de los artículos seleccionados en cuanto a la ejecución de la cadena de búsqueda. Se puede observar que existe una recurrencia de publicaciones del dominio de las ramas de “component-based Software”, “componentes”, “cots”, “ciclos de vida y frameworks” forman parte de esta lista

Tabla 14 Publicaciones encontradas relacionado al tema

ID	Título	Cantidad
1	A Study of Component Based Software System Metrics	1
2	Towards systematic software reuse of GIS: Insights from a case study	1
3	A cloud service for COTS component-based architectures	1
4	The Framework and Its Implementation for Managing Component-based Software Evolution	1
5	Component based software engineering: quality assurance models, metrics	1
6	A formal approach for managing component-based architecture evolution	1
7	An Analytical Study of Component-Based Life- Cycle Models: A Survey	1

8	Usability-Focused Architectural Design for Graphical User Interface Components	1
9	Software Component Retrieval Using Genetic Algorithms	1
10	A Method to Generate Reusable Safety Case Argument-Fragments from Compositional Safety Analysis	1
11	Component-based Software Update Process in Collaborative Software Development	1
12	How Can Optimization Models Support the Maintenance of Component-Based Software?	1
13	Software Behavior Description of Real-Time Embedded Systems in Component Based Software Development	1
14	New Method to Find the Maximum Number of Faults by Analyzing Reliability and Reusability in Component-Based Software	1
15	Round Tripping in Component Based Software Development	1
16	Life Cycles for Component-Based Software Development Syed	1
17	Toward an Architecture-Based Method for Selecting Composer Components to Make Software Product Line	1
18	A flexible data acquisition system for storing the interactions on mashup user interfaces	1

1. Preguntas de investigación 1 (PI-1)

¿Qué métodos de trabajo son adecuados para una arquitectura de software basada en componentes?

A través de los análisis de los artículos se pudo encontrar diferentes métodos de trabajo aplicados para arquitecturas de software basado en componentes. Dentro de los métodos de trabajo encontrados los más relevantes son:

Método Halstead para calcular el número máximo de fallas a través de la investigación de fiabilidad y reutilización del software basado en componente. Este nuevo método es muy efectivo para minimizar el costo y el tiempo en el desarrollo de software basado en componentes. Este método encuentra efectos en el proceso de reutilización, en las fases de análisis de requisitos, diseño, codificación y simultáneamente en la fiabilidad del código del programa, así mismo también se enfoca en factores principales como la confiabilidad y la reutilización. Si los factores de confiabilidad y reutilización se analizan antes de la fase de prueba, entonces sería beneficioso tanto para el desarrollador como para el cliente. Este método propuesto se aplica antes de la fase de prueba del ciclo de vida de desarrollo, de acuerdo con Halsted Software Science. Estas medidas son:

Halstead usa medidas primitivas para desarrollar expresiones para la longitud global del software, volumen mínimo potencial para un algoritmo, volumen real [28], es decir, el número de bits requeridos para especificar un software. Dichas medidas se listan a continuación:

- n1: El número de operadores diferentes que usa el software.
- n2: El número de operandos diferentes que usa el software.
- N1: El número total de veces que aparece el operador
- N2 El número total de veces que aparece el operando

Halstead expone una longitud se puede estimar como: N= n1 log1 + n2log2 y el volumen del programa es definido como: V= N log2(n1+n2). Es necesario tener en cuenta que V variarà con el lenguaje de programación.

Modelo de dominio. Agiliza la resolución de problemas a través de un modelo conceptual basándose en los temas relacionados con el problema en específico, en otras palabras, es modelar lo necesario para comprender cómo funciona los componentes, por consiguiente, permite reconocer requisitos para un componente, además, proporciona una visión más estructurada. En la figura 7 se presenta un ejemplo del modelo mencionado en el que se analiza los requerimientos del sistema, de tal forma que se obtiene un diagrama, esto permite reconocer que requisitos son necesarios para ser un componente. Por ejemplo, la entidad médica puede utilizarse en cualquier sistema tomando en cuenta sus atributos que lo conforman ya que en cualquier hospital necesariamente se tiene que tener un registro de sus trabajadores. Este modelo simplemente ayuda agilizando a reconocer componentes de un sistema [29].

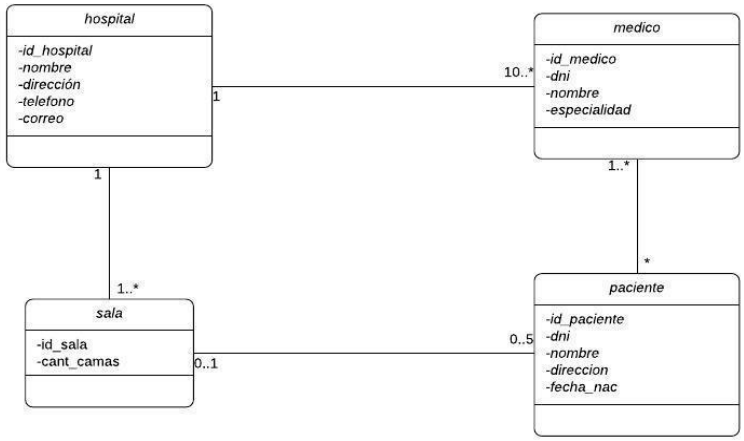


Figura 7. Ejemplo de modelo de dominio

Modelo de calidad de componente (CQM):

Este modelo fue propuesto por Joaquina Martin-Albo, Manuel F. Bertoa, Coral Calero; en donde evalúan la calidad de los componentes en cuatro dimensiones y se centran principalmente en las métricas de usabilidad. Además, se propone cuatro dimensiones que deben ser considerado en la evaluación de la calidad del componente que son: características de calidad (distinguir entre Calidad externa / interna y calidad de uso), Granularidad, Visibilidad de un sistema completo y aislado componentes [30].

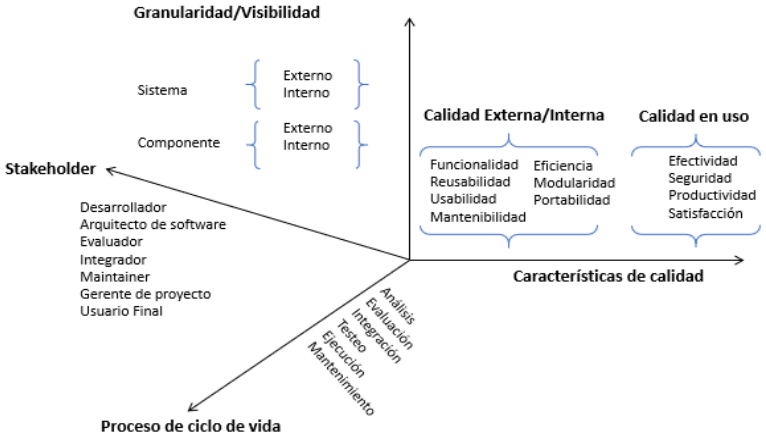


Figura 8. Modelo de calidad de componentes

En la figura 8 se observa un de calidad de componentes que está dividido en dos subcategorías para distinguir entre la calidad externa / interna y la calidad en el uso de un sistema basado en componente, es decir la calidad externa se refiere a la ejecución del software mientras que la calidad interna se refiere a la perspectiva intermedia del producto.

Modelo arquitectónico Dedal:

Un modelo que apunta al desarrollo de la reutilización de componentes mediante un ciclo de vida. La idea principal es construir una arquitectura concreta compuesta por componentes almacenados en un repositorio para facilitar las decisiones de diseño y construcción de un proyecto de software [26]. Este modelo se propone tres pasos para el ciclo de vida de componentes:

- **Especificación**: Abarca el enfoque y arquitectura abstracta para cumplir con los requisitos del software a futuro, en otras palabras, primero se debería tener una idea de la arquitectura, una arquitectura imaginada por el arquitecto de software en el que contemple la escalabilidad, desempeño, usabilidad, modificabilidad, facilidad de prueba y seguridad. Después de analizar el enfoque se plantea los roles y especificaciones de los componentes con el motivo de ensamblarlos y que además sean compatibles ante cualquier software o componente externo [26].

- **Implementación**: Abarca toda la configuración y desarrollo de los componentes. En primer lugar, se identifica las clases que pueda tener un componente, sea clases primitivas o compuestas. Las clases primitivas encapsulan todo el código ejecutable mientras que las clases compuestas encapsula una configuración interna, es decir, encapsula un conjunto de clases conectados sea primitivo o compuestos. Finalmente son almacenados en un repositorio de componentes para después ser ensamblados en un software [26].

- **Despliegue**: En primer lugar, se crea un modelo para identificar las instancias de los componentes que serán ensamblados dependiendo del tiempo de ejecución, las clases primitivas o compuestas, los roles y especificaciones de los componentes. Cada instancia de componente tiene un estado inicial y un estado final por medio de una lista de atributos valorados [26].

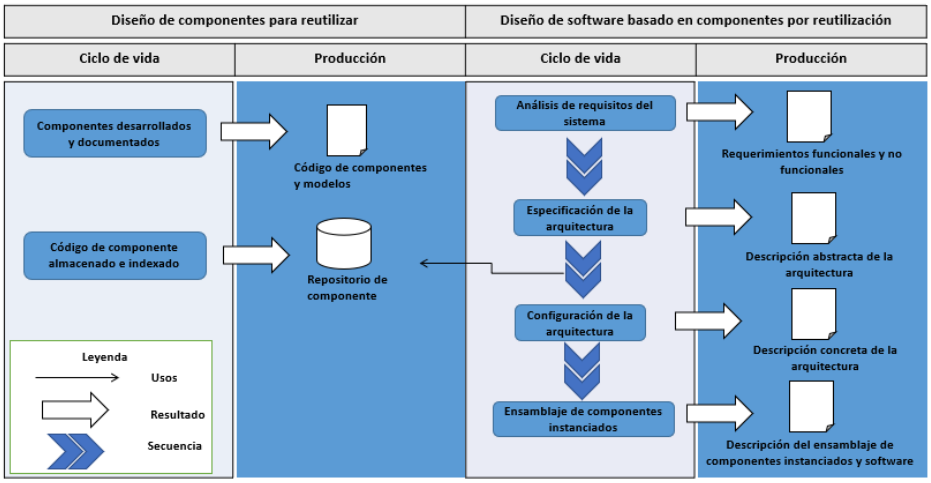


Figura 9. Proceso de desarrollo centrado en la reutilización [31]

El proceso de la figura 9 se muestra 2 ciclos de vida de desarrollo de software. El ciclo llamado "Diseño de componentes para reutilizar" se centra en la creación de los componentes, primero se desarrolla los componentes y se documenta su funcionalidad, una vez desarrollado los componentes, se integra a un repositorio para un uso posterior. En siguiente ciclo de vida llamado "Diseño de software basado en componentes por reutilización" es lo que propone el modelo arquitectónico Dedal. En primer lugar, se analiza los requerimientos del sistema y se documenta sus requerimientos funcionales y no funcionales. En la especificación de la arquitectura se describe la arquitectura que tendrá a futuro y las especificaciones de los componentes (relación, funcionalidad, modificabilidad, etc.), posteriormente se configura la arquitectura tomando en cuenta las especificaciones del proceso anterior, luego se ensambla los componentes que se encuentran en el repositorio.

2. Preguntas de investigación 2 (PI-2)

¿Qué herramientas de modelado de software, construcción e implementación existen para una arquitectura de software basada en componentes?

De acuerdo a la extracción de los artículos se puede observar que existen diferentes herramientas de trabajo aplicados para arquitecturas de software basado en componentes. Dentro de las herramientas que se han encontrado, los más relevantes son:

1.- **Lenguaje de dominio específico** (**DSL**):

Es un lenguaje especialmente dedicado a resolver un problema en particular, se diferencia de los demás lenguajes de programación como java, php, C#, etc porque solo funciona para resolver problemas en el dominio que se le asignó. Por ejemplo, fórmulas de las planillas de cálculo, comandos en un gestor de paquetes, comando para crear un objeto en una base de datos. Esta herramienta es indispensable si se desea trabajar en base a componentes.

Jesús Vallecillos y Javier Criado utilizaron un diseño inspirado en ingeniería basado en modelos (**MDE**) en el que especifican que un modelo arquitectónico concreto consiste en un conjunto de componentes individuales (ConcreteComponent) y un conjunto de relaciones (Relations) conecta dos o más componentes simultáneamente. Cada componente tiene un tipo (ComponentType), existe componente contenedor, componente funcional y el componente usuario-interacción. Los componentes contenedores identifican si tiene relación con otros componentes y los componentes funcionales seria la funcionalidad de los requerimientos sin la interacción del usuario. El componente usuario-interacción se utiliza solamente para mostrar y obtener la información a través del usuario. También existe otro tipo de componente llamado componente normal que es la unión del componente funcional y componente usuario-interacción. Cada componente posee un puerto que permite la conexión entre varios componentes [27].

2.- **Diagrama de componentes StartUML:**

UML es un lenguaje gráfico que sirve para visualizar, especificar, construir y documentar un sistema; fue presentado en 1999 por Fischman para estimar el tamaño del software a partir de su diseño de Lenguaje de modelado unificado. UML es la presentación y visualización de arquitectura de software; hay tres categorías de diagramas UML que describen el sistema de software desde diferentes perspectivas: estructural, dinámica y gestión de modelos [32].

En la tabla 15 se observa las tres categorías de diagramas UML, a través de ello se puede identificar que la herramienta identificada a través de la cadena de búsqueda pertenece a la categoría estructural que abarca diagrama de componentes.

Tabla 15 Categorías de diagramas UML

Categoría	Diagrama
Estructural	- Use el diagrama del caso - Diagrama de componentes - Diagrama de clase - Diagrama de implementación
dinámica	- Diagrama de secuencia - Diagrama de colaboración - Diagrama de actividad - Diagrama de transición de estado
Gestión de modelos	- Diagrama de clases que describe paquetes, subsistemas y modelos.

StartUML es una herramienta de modelado de software de código abierto que admite UML (Unified Modeling Language). proporciona once tipos diferentes de diagrama. Apoya activamente el enfoque MDA (Model Driven Architecture) y permitir generar código para múltiples lenguajes. La herramienta StartUML se usa para diagramar componentes de software, dicha herramienta ayuda mucho en la fase de diseño de software principalmente enfocado en el modelado de componentes.

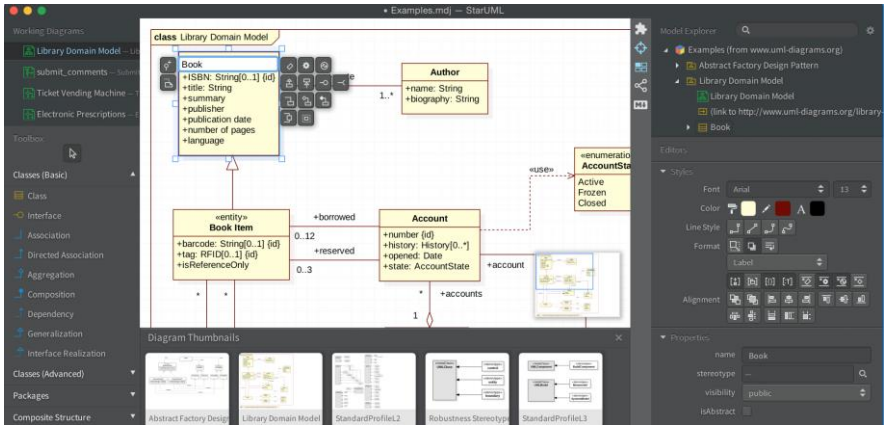


Figura 10. IDE Start UML Modelado de componentes

En la figura 8 se puede observar un modelador de componentes de software que trabaja con plataformas macOS, Windows y Linux. Este modelador describe varios modelos de componentes, puesto que no es suficiente describir un sistema de software desde un solo punto de vista. Como, por ejemplo, se requiere construir un sistema de facturación utilizando el enfoque de arquitectura de software basado en componentes, para ello se necesita realizar varios modelos: 1) modelo que cubra el componente de autenticación, 2) Componente de clientes, etc.

5 Conclusiones

En este artículo de revisión se muestran los resultados de una revisión sistemática de la literatura realizada a 18 artículos académicos encontrados en 3 bases de datos indexadas de gran relevancia en el ámbito académico y científico. Por otra parte, también se presentó el análisis bibliométrico de clasificación de los estudios por año de publicación, donde hubo un incremento en el año 2008, luego en el año 2016 y 2017 por lo que indica que existe un interés constante en la optimización de un software basado en componentes, así mismo se pudo notar que la mayor cantidad de artículos fueron en idioma inglés. Por otra parte, las preguntas de investigación determinaron los métodos y herramientas que influyen a una arquitectura basada en componentes; como por ejemplo tenemos (CQM) para evaluar la calidad de los componentes en base a cuatro dimensiones 1) Calidad de software 2) Visibilidad y granularidad 3) Interesados del proyecto(stakeholders) y 4) procesos de ciclo de vida; a diferencias del modelo arquitectónico DEDAL que apunta al desarrollo de la reutilización de componentes mediante un ciclo de vida. En cuanto a las herramientas encontradas mediante la cadena de búsqueda se identificó StartUML una herramienta para el modelado de componentes que viene a ser de mucha ayuda en la fase de diseño de software, seguidamente se identificó la herramienta de Lenguaje de dominio específico (DSL) junto con MDE (ingeniería basado en modelos) para poder diagramar un modelo arquitectónico que ayudar en la fase de especificación de arquitectura. Por otra parte, se pretende resaltar la importancia de los componentes en un software, no solo por el producto, sino también por la reusabilidad de componentes prefabricados, sobre todo en la optimización de productividad en el desarrollo de software.

Finalmente, se espera que, por la arquitectura de componentes, surjan temas de interés sobre la mantenibilidad y la gestión de componentes, temas que serán de mucha importancia para poder tener un mejor control de estos componentes a la hora de implementarlos con el fin de hacerlo comercial (COTS).

Referencias

[1] M. Ramachandran and G. Singh Jamnal, “Developing Reusable .NET Software Components,” 2014.

[2] R. G. C. Rocha *et al.*, “Collaboration Models in Distributed Software Development: a Systematic Review,” 2011.

[3] S. A. Fahmi and H. J. Choi, “Life cycles for component-based software development,” in *Proceedings - 8th IEEE International Conference on Computer and Information Technology Workshops, CIT Workshops 2008*, 2008, pp. 637–642.

[4] T. Longye, W. Zhijian, and F. Yukui, “A framework for component based software flexible design,” in *Proceedings of 2009 4th International Conference on Computer Science and Education, ICCSE 2009*, 2009, pp. 937–940.

[5] Ali Arshad ; Muhammad Usman, “Security at software architecture level: A systematic mapping study,” 2011.

[6] H. Venkitachalam, J. Richenhagen, A. Schlosser, and T. Tasky, “Metrics for verification and validation of architecture in powertrain software development,” in *WASA 2015 - Proceedings of the 2015 ACM Workshop on Automotive Software Architecture, Part of CompArch 2015*, 2015, pp. 27–34.

[7] D. Kumar and M. Kumari, “Component based software engineering: Quality assurance models, metrics,” in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2015*, 2015.

[8] I. Crnković, S. Sentilles, V. Aneta, and M. R. V. Chaudron, “A classification framework for software component models,” *IEEE Trans. Softw. Eng.*, vol. 37, no. 5, pp. 593–615, 2011.

[9] T. Vale, I. Crnkovic, E. S. De Almeida, P. A. D. M. Silveira Neto, Y. C. Cavalcanti, and S. R. D. L. Meira, “Twenty-eight years of component-based software engineering,” *J. Syst. Softw.*, vol. 111, pp. 128–148, Jan. 2016.

[10] L. Zhong, J. Xia, and X. Huang, “The Framework and Its Implementation for Managing Component-Based Software Evolution,” in *Proceedings - 2016 3rd International Conference on Information Science and Control Engineering, ICISCE 2016*, 2016, pp. 711–715.

[11] V. Cortellessa and P. Potena, “How can optimization models support the maintenance of component-based software?,” in *Proceedings - 1st International Symposium on Search Based Software Engineering, SSBSE 2009*, 2009, pp. 97–100.

[12] Luis Felipe Fernandez, “Arquitectura de software,” 2006.

[13] D. E. Perry, T. B. Laboratories, M. Hill, and A. L. Wolf, “Foundations for the Study of

Software Architecture," vol. 17, no. 4, 1992.

[14] N. Medvidovic and R. N. Taylor, "Software Architecture : Foundations , Theory , and Practice," *Africa (Lond).*, pp. 471–472, 2010.

[15] P. Kruchten, "What do software architects really do?," *J. Syst. Softw.*, vol. 81, no. 12, pp. 2413–2416, Dec. 2008.

[16] M. Razavian, B. Paech, and A. Tang, "Empirical research for software architecture decision making: An analysis," *J. Syst. Softw.*, vol. 149, pp. 360–381, Mar. 2019.

[17] N. M. Villegas, G. Tamura, and H. A. Müller, "Architecting Software Systems for Runtime Self-Adaptation," in *Managing Trade-Offs in Adaptable Software Architectures*, Elsevier, 2017, pp. 17–43.

[18] A. Sharma, M. Kumar, and S. Agarwal, "A Complete Survey on Software Architectural Styles and Patterns," *Procedia Comput. Sci.*, vol. 70, pp. 16–28, 2015.

[19] F. J. Wang and F. Fahmi, "Constructing a service software with microservices," in *Proceedings - 2018 IEEE World Congress on Services, SERVICES 2018*, 2018, pp. 33–34.

[20] D. Alonso, J. Á. Pastor, P. Sánchez, B. Álvarez, and C. Vicente-Chicote, "Generación automática de software para sistemas de tiempo real: Un enfoque basado en componentes, modelos y frameworks," *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 9, no. 2, pp. 170–181, 2012.

[21] A. Buccella, A. Cechich, M. Arias, M. Pol'la, M. del S. Doldan, and E. Morsan, "Towards systematic software reuse of GIS: Insights from a case study," *Comput. Geosci.*, vol. 54, pp. 9–20, Apr. 2013.

[22] J. Sharma, A. Kumar, and Kavita, "A Design Based New Reusable Software Process Model for Component Based Development Environment," *Procedia Comput. Sci.*, vol. 85, no. Cms, pp. 922–928, 2016.

[23] K. K. Lau and Z. Wang, "Software component models," *IEEE Trans. Softw. Eng.*, vol. 33, no. 10, pp. 709–724, Oct. 2007.

[24] I. Zafar, A. Shaheen, A. K. Nazir, B. Maqbool, W. H. Butt, and J. Zeb, "Why Pakistani Software Companies don't use Best Practices for Requirement Engineering Processes," *2018 IEEE 9th Annu. Inf. Technol. Electron. Mob. Commun. Conf.*, pp. 996–999, 2018.

[25] H. G. Min, J. Y. Lee, S. A. Kim, and S. D. Kim, "An effective method to design CBD components in Enterprise JavaBeans (EJB)," in *Proceedings - Fourth International Conference on Software Engineering Research, Management and Applications, SERA 2006*, 2006, pp. 49–56.

[26] D. Negi, Y. S. Chauhan, P. Dimri, and A. Harbola, “An Analytical Study of Component-Based Life Cycle Models: A Survey,” in *Proceedings - 2015 International Conference on Computational Intelligence and Communication Networks, CICN 2015*, 2016, pp. 746–750.

[27] J. Vallecillos, J. Criado, N. Padilla, and L. Iribarne, “A cloud service for COTS component-based architectures,” *Comput. Stand. Interfaces*, vol. 48, pp. 198–216, Nov. 2016.

[28] D. Panwar and P. Tomar, “New method to find the maximum number of faults by analyzing reliability and reusability in component-based software,” in *TISC 2011 - Proceedings of the 3rd International Conference on Trendz in Information Sciences and Computing*, 2011, pp. 164–168.

[29] A. J. Fernández-García, L. Iribarne, A. Corral, J. Criado, and J. Z. Wang, “A flexible data acquisition system for storing the interactions on mashup user interfaces,” *Comput. Stand. Interfaces*, vol. 59, pp. 10–34, Aug. 2018.

[30] S. Patel and J. Kaur, “A study of component based software system metrics,” in *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*, 2017, pp. 824–828.

[31] A. Mokni, C. Urtado, S. Vauttier, M. Huchard, and H. Y. Zhang, “A formal approach for managing component-based architecture evolution,” in *Science of Computer Programming*, 2016, vol. 127, pp. 24–49.

[32] Y. Chen, B. W. Boehm, R. Madachy, and R. Valerdi, “An Empirical Study of eServices Product UML Sizing Metrics,” 2004.

Apéndice

A. Artículos seleccionados

ID	Biblioteca	Título	Autor	Año	Tipo
1	Ieeexplore	A Study of Component Based Software System Metrics	Sakshi Patel, Jagdeep Kaur	2017	Conferencia
2	ScienceDirect	Towards systematic software reuse of GIS: Insights from a case study	Agustina Buccella, Alejandra Cechich, Maximiliano Arias, Matias Pol'la, Maria del Socorro Doldan, Enrique Morsan	2013	Artículo de jornada
3	ScienceDirect	A cloud service for COTS component-based architectures	Jesús Vallecillos, Javier Criado, Nicolás Padilla, Luis Iribarne	2015	Artículo de jornada
4	Ieeexplore	The Framework and Its Implementation for Managing Component-based Software Evolution	linhui Zhong,jing Xia,xiaoming Huang	2016	Conferencia
5	Ieeexplore	Component based software engineering: quality assurance models, metrics	Deepak Kumar, Meghna Kumari	2015	Conferencia
6	ScienceDirect	A formal approach for managing component-based architecture evolution	Mokni, Abderrahman Urtado, Christelle Vauttier, Sylvain Huchard, Marianne Zhang, Huaxi Yulin	2016	Artículo de jornada
7	Ieeexplore	An Analytical Study of Component-Based Life- Cycle Models: A Survey	Negi, Deepti Chauhan, Yashwant Singh Dimri, Priti Harbola, Aditya	2016	Conferencia
8	Ieeexplore	Usability-Focused Architectural Design for Graphical User Interface Components	Stephan Bode, Matthias Riebisch	2008	Conferencia
9	Ieeexplore	Software Component Retrieval Using Genetic Algorithms	Dixit, Anurag Saxena, P. C.	2009	Conferencia
10	ScienceDirect	A Method to Generate Reusable Safety Case Argument-Fragments from Compositional Safety Analysis	Šljivo, Irfan Gallina, Barbara Carlson, Jan Hansson, Hans Puri, Stefano	2017	Artículo de jornada
11	Ieeexplore	Component-based Software Update Process in Collaborative Software Development	Nguyen, Tien N.	2008	Conferencia
12	Ieeexplore	How Can Optimization Models	Cortellessa, Vittorio	2009	Conferencia

		Support the Maintenance of Component-Based Software?	Potena, Pasqualina		
13	Ieeexplore	Software Behavior Description of Real-Time Embedded Systems in Component Based Software Development	Kim, Ji Eun Kapoor, Rahul Herrmann, Martin Haerdtlein, Jochen Grzeschniok, Franz Lutz, Peter	2008	Conferencia
14	Ieeexplore	New Method to Find the Maximum Number of Faults by Analyzing Reliability and Reusability in Component-Based Software	Panwar, Deepak Tomar, Pradeep	2011	Conferencia
15	Ieeexplore	Round Tripping in Component Based Software Development	Wautelet, Yves Kiv, Sodany Tran, Vi Kolp, Manuel	2010	Conferencia
16	Ieeexplore	Life Cycles for Component-Based Software Development Syed	Fahmi, Syed Ahsan Choi, Ho Jin	2008	Conferencia
17	Ieeexplore	Toward an Architecture-Based Method for Selecting Composer Components to Make Software Product Line	Tanhaei, Mohammad Moaven, Shahrouz Habibi, Jafar	2010	Conferencia
18	ScienceDirect	A flexible data acquisition system for storing the interactions on mashup user interfaces	Fernández-García, Antonio Jesús Iribarne, Luis Corral, Antonio Criado, Javier Wang, James Z.	2018	Artículo de jornada

B. Formularios de extracción

A continuación, se presenta la información extraída de todos los artículos seleccionados que contribuyeron a responder las preguntas de investigación.

Criterio	**Detalle**	**Relevancia**
Identificador	1	-
Fuente	IEEExplore	PB-I
Título	A Study of Component Based Software System Metrics	PI-2
Autores	Sakshi Patel, Jagdeep Kaur	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc	PBI-3
Año de publicación	2017	PBI-2
Tipo de publicación	Conferencia	PBI-2
Objetivo de análisis	Estudiar métricas basada en componentes para proporcionar reutilización y disminuir el costo y el tiempo de desarrollo.	PI-I

Criterio	Detalle	Relevancia
Identificador	2	-
Fuente	ScienceDirect	PB-I
Título	Towards systematic software reuse of GIS: Insights from a case study	PI-2
Autores	Agustina Buccella, Alejandra Cechich, Maximiliano Arias, Matias Pol'la, Maria del Socorro Doldan, Enrique Morsan,	PBI-1
Publicación	Computers and Geosciences	PBI-3
Año de publicación	2013	PBI-2
Tipo de publicación	Artículo de jornada	PBI-2
Objetivo de análisis	Un enfoque orientado al desarrollo de componentes comerciales basado en la experiencia recopilada de un estudio de caso	PI-I

Criterio	Detalle	Relevancia
Identificador	3	-
Fuente	ScienceDirect	P-I
Título	A cloud service for COTS component-based architectures	PI-1
Autores	Jesús Vallecillos, Javier Criado, Nicolás Padilla, Luis Iribarne	PBI-1
Publicación	Computer Standards and Interfaces	PBI-1
Año de publicación	2016	PBI-1
Tipo de publicación	Artículo de jornada	
Objetivo de análisis	Una solución de infraestructura basada en el uso de servicios web para ser administradas por componentes	PI-1

Criterio	Detalle	Relevancia
Identificador	4	-
Fuente	IEEExplore	P-I
Título	The Framework and Its Implementation for Managing Component-based Software Evolution	PI-1
Autores	linhui Zhong,jing Xia,xiaoming Huang	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2016	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Prototipo de sistema de soporte desde la perspectiva técnica y de gestión de procesos para los CBSE	PI-1

Criterio	Detalle	Relevancia
Identificador	5	-
Fuente	IEEExplore	P-I
Título	Component based software engineering: quality assurance models, metrics	PI-1
Autores	Deepak Kumar, Meghna Kumari	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2015	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Presentar un modelo de aseguramiento de calidad de CBM	PI-1

Criterio	Detalle	Relevancia
Identificador	6	-
Fuente	ScienceDirect	P-I
Título	A formal approach for managing component-based architecture evolution	PI-1
Autores	Abderrahman Mokni, Christelle Urtado, Sylvain Vauttier, Marianne Huchard, Huaxi YulinZhang	PBI-1
Publicación	Elsevier B.V.	PBI-1
Año de publicación	2016	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Presentar un modelo de gestión de evolución que genera planes de evolución de acuerdo con una solicitud de cambio de arquitectura	PI-1

Criterio	Detalle	Relevancia
Identificador	7	-
Fuente	IEEExplore	P-I
Título	An Analytical Study of Component-Based Life- Cycle Models: A Survey	PI-1
Autores	Deepti Negi, Yashwant Singh Chauhan, Priti Dimri, Aditya Harbola,	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2016	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Comparación de los modelos de ciclo de vida basados en componentes V, Y, W, X, Knot, New Era, Elicit y Elite Plus.	PI-1

Criterio	Detalle	Relevancia
Identificador	8	-
Fuente	IEEExplore	P-I
Título	Usability-Focused Architectural Design for Graphical User Interface Components	PI-1
Autores	Stephan Bode, Matthias Riebisch	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2008	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	. Presentar una metodología que extiende un método de diseño arquitectónico de software basado en categorías al integrar enfoques HCI.	PI-1

Criterio	Detalle	Relevancia
Identificador	9	-
Fuente	IEEExplore	P-I
Título	Software Component Retrieval Using Genetic Algorithms	PI-1
Autores	Anurag Dixit, P. C. Saxena	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2009	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Desarrollar un enfoque basado en algoritmos para el componente de selección	PI-1

Criterio	Detalle	Relevancia
Identificador	10	-
Fuente	ScienceDirect	P-I
Título	A Method to Generate Reusable Safety Case Argument-Fragments from Compositional Safety Analysis	PI-1
Autores	Irfan Šljivo, Barbara Gallina, Jan Carlson, Hans Hansson, Stefano Puri,	PBI-1
Publicación	Elsevier Inc.	PBI-1
Año de publicación	2017	PBI-1
Tipo de publicación	Artículo de jornada	
Objetivo de análisis	Método para generar casos reusables a partir de un análisis de seguridad composicional	PI-1

Criterio	Detalle	Relevancia
Identificador	11	-
Fuente	IEEExplore	P-I
Título	Component-based Software Update Process in Collaborative Software Development	PI-1
Autores	Tien N Nguyen	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2008	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Proporcionar una base para la implementación de actualización de software basada en componentes	PI-1

Criterio	Detalle	Relevancia
Identificador	12	-
Fuente	IEEExplore	P-I
Título	How Can Optimization Models Support the Maintenance of Component-Based Software?	PI-1
Autores	Vittorio Cortellessa, Pasqualina Potena	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2009	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Presentar técnicas de optmización para gestionar el problema de fallas durante el mantenimiento por componentes	PI-1

Criterio	Detalle	Relevancia
Identificador	13	-
Fuente	IEEExplore	P-I
Título	Software Behavior Description of Real-Time Embedded Systems in Component Based Software Development	PI-1
Autores	Ji Eun Kim, Rahul Kapoor, Martin Herrmann, Jochen Haerdtlein, Franz Grzeschniok, Peter Lutz,	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2008	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Presentar concepto y estudio de caso de "signal flows" y "mode dependent signal flows". Para así obtener información crucial sobre el comportamiento del software para sistemas integrados en tiempo real a nivel de componente mediante	PI-1

Criterio	Detalle	Relevancia
Identificador	14	-
Fuente	IEEExplore	P-I
Título	New Method to Find the Maximum Number of Faults by Analyzing Reliability and Reusability in Component-Based Software	PI-1
Autores	Deepak Panwar, Pradeep Tomar	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2011	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Calcular el número máximo de fallas a través de la investigación de la confiabilidad y la reutilización del software basado en componentes.	PI-1

Criterio	Detalle	Relevancia
Identificador	15	-
Fuente	IEEExplore	P-I
Título	Round Tripping in Component Based Software Development	PI-1
Autores	Yves Wautelet, Sodany Kiv, Vi Tran, Manuel Kolp	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2010	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Proponer un proceso de desarrollo basado en componentes que utiliza un procedimiento de selección de componentes de dos niveles con bucles de retroalimentación.	PI-1

Criterio	Detalle	Relevancia
Identificador	16	-
Fuente	IEEExplore	P-I
Título	Life Cycles for Component-Based Software Development Syed	PI-1
Autores	Syed Ahsan Fahmi, Ho Jin Choi	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2008	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Comparar los desafíos que conlleva el desarrollo basado en componentes	PI-1

Criterio	Detalle	Relevancia
Identificador	17	-
Fuente	IEEExplore	P-I
Título	Toward an Architecture-Based Method for Selecting Composer Components to Make Software Product Line	PI-1
Autores	Mohammad Tanhaei, Shahrouz Moaven, Jafar Habibi,	PBI-1
Publicación	Institute of Electrical and Electronics Engineers Inc.	PBI-1
Año de publicación	2010	PBI-1
Tipo de publicación	Conferencia	
Objetivo de análisis	Presentar método para gestionar y controlar las complejidades del problema de selección de componentes	PI-1

Criterio	Detalle	Relevancia
Identificador	18	-
Fuente	ScienceDirect	P-I
Título	A flexible data acquisition system for storing the interactions on mashup user interfaces	PI-1
Autores	Antonio Jesús Fernández-García, Luis Iribarne, Antonio Corral, Javier Criado, James Z Wang	PBI-1
Publicación	Elsevier B.V.	PBI-1
Año de publicación	2018	PBI-1
Tipo de publicación	Artículo de jornada	
Objetivo de análisis	Presentar un sistema flexible de adquisición de datos capaz de capturar las interacciones humano-computadora realizadas por los usuarios a través de interfaces mashup (usuario) con el objetivo de almacenarlas en una base de datos relacional basado en componentes.	PI-1