Información Tecnológica
Vol. 27(4), 193-202 (2016)
doi: 10.4067/S0718-07642016000400021

Tipificación de Dominios de Requerimientos para la Aplicación de Patrones Arquitectónicos

Johanna M. Suárez[(1)]* y Luz E. Gutiérrez[(2)]
(1) Unidades Tecnológicas de Santander, Calle de los Estudiantes N° 9-82 Ciudadela Real de Minas, Santander, Bucaramanga - Colombia (e-mail: jsuarez@correo.uts.edu.co)
(2) Universidad Santo Tomás, Seccional Tunja, Sede Centro: Calle. 19 Nº 11 – 64, Tunja, Boyacá – Colombia (e-mail: decsistemas@ustatunja.edu.co)

* Autor a quien debe ser dirigida la correspondencia

Recibido Sep. 19, 2015; Aceptado Nov. 18, 2015; Versión final Feb. 20, 2016, Publicado Ago. 2016

Resumen

El presente artículo describe los resultados de un trabajo de investigación desarrollado para apoyar la fase de diseño de arquitectura de software de un proyecto. Como principal producto se obtuvo un dominio de requerimientos que agrupa los elementos comunes en proyectos de desarrollo web que promueven la integración de plataformas y los ecosistemas digitales. Este dominio se utilizó como insumo para relacionar los patrones arquitectónicos que son utilizados actualmente y de esta forma, definir un recurso que podría ser útil en cualquier proyecto de desarrollo para la selección del patrón más adecuado. Para la ejecución del proyecto se realizaron actividades de selección de proyectos de desarrollo de software, identificación de requisitos funcionales y no funcionales y selección de patrones arquitectónicos de referencia. También se realizó la validación del dominio de requerimientos para corroborar que los patrones arquitectónicos asociados, representaban una opción pertinente para el requerimiento.

Palabras clave: patrón de arquitectura; dominio de requerimientos; arquitectura software; desarrollo software

Domain Classification Requirements for Application of Architectural Patterns

Abstract

This article describes the results of a research developed to support the design phase of software architecture of a project, as the main product a domain requirement that groups the common elements in web development projects that promote the integration of platforms was obtained and digital ecosystems. This domain was used as an input to relate architectural patterns that are currently used and thus define a resource that could be useful in any development project for the selection of the most suitable pattern. For project implementation activities were carried out: selection of software development projects; identifying functional and nonfunctional requirements; selection of architectural benchmarks and domain validation requirements to substantiate that the architectural patterns associated, represented an appropriate option for the requirement.

Keywords: architectural pattern; domain requirements; software architecture; software development

INTRODUCCIÓN

La ingeniería de software es aplicable a la construcción de software en todas las áreas de la computación y brinda un conjunto de herramientas y técnicas para orientar procesos de desarrollo, todo este conocimiento es de carácter formal y es transmitido a los estudiantes de a través de los currículos como parte de su formación (IEEE y ACM , 2013). No obstante, algunos de estos conceptos transmitidos son netamente teóricos y difíciles de llevar a la práctica tanto por un estudiante como por un profesional. Una muestra que evidencia lo anteriormente expuesto se puede encontrar en el tema de Arquitectura Software -AS-, Patidar y Suman (2015) la definen como compleja en sí mismas, afirman que a veces no es fácil implementar en la práctica las decisiones de diseño incorporadas en la arquitectura.

La industria de software ha entendido la verdadera importancia de la -AS-, por ende, es fácil encontrar en los actuales desarrollos, recurso humano con el perfil de arquitectos de software encargados de definir los artefactos y los esquemas de integración de los sistemas a desarrollar (Lago y Vliet, 2005). Un aspecto fundamental que ha incidido en la importancia de la -AS- es la integración de múltiples sistemas (Dabous, Rabhi, y Yu, 2004), puesto que hoy en día lo sistemas no se desarrollan para estar aislados, deben ser parte de un todo y los métodos y protocolos a seguir para interconectarlos debe ser definidos antes de diseñar y construir el sistema. En ese proceso de integración y en el marco de la calidad de los sistemas surge el tema de los atributos de calidad (Bass et al., 2003), éstos son aspectos no funcionales que deben ser tenidos en cuenta en el desarrollo de sistemas software y que requieren de un análisis exhaustivo que permita encontrar una solución eficaz y eficiente que lógicamente optimice los recursos tanto del proyecto como del cliente.

Ahora bien, la ingeniería del software dispone de los patrones para dar soluciones a problemas comunes y específicos en las diferentes fases de un proceso de desarrollo, en el diseño de software se pueden encontrar los patrones de diseño propuestos por GOF (Gamma et al., 1994), los cuales han sido utilizados durante los últimos años con un verdadero éxito en procesos de reutilización, tanto de diseño cómo de código (Ampatzoglou et al., 2015). En el campo de la -AS- también se pueden encontrar patrones que permiten brindar soluciones a la hora de construir la base de un sistema software. A pesar de esto, en este campo el éxito no ha sido tan rotundo como lo han sido los patrones de diseño (Klein, 2016), de hecho, un arquitecto software con experiencia selecciona una solución soportado en su experticia más no, en el conocimiento explicito que pueda llegar a tener sobre patrones arquitectónicos, ¿por qué razón?, en el marco de este estudio, una respuesta arriesgada pero que encaja perfectamente en este aspecto es: *los patrones arquitectónicos carecen de un catálogo que explique paso a paso la funcionalidad y la relación con los atributos de calidad*. En este escenario surgen múltiples oportunidades para abordar problemas encaminados a apoyar proyectos que utilicen patrones arquitectónicos que permitan construir una base sólida para el desarrollo de sistemas software. Teniendo en cuenta lo anterior, surgió la siguiente pregunta de investigación que permitió obtener el resultado presentado en este artículo: ¿Es posible definir una tipificación de dominios de requerimientos para la aplicación de patrones arquitectónicos en un proceso de desarrollo software específico?

Para seleccionar un patrón arquitectónico probado y validado se requiere experiencia (Klein, 2016), no solo en el área de diseño y desarrollo, sino también en la fase de requerimientos tanto funcionales como no funcionales, esto hace del arquitecto de software una persona necesariamente con mucha experiencia en el campo de desarrollo software, con base en esta premisa, surgen otros interrogantes a resolver, ¿*Puede la tipificación de patrones de arquitectura soportar un método que oriente la selección de patrones arquitectónicos para acortar la brecha entre ingenieros en formación y profesionales con amplia experiencia y trayectoria*?. Indiscutiblemente la oportunidad existe y está orientada al mejoramiento de la formación de los profesionales en el área de sistemas e informática, al igual que a la necesidad de tener herramientas que permitan guiar un proceso que a la fecha es difícil de realizar y en la mayoría de los casos solo lo ejecutan expertos con una trayectoria de muchos años en el campo de desarrollo.

Es claro que las empresas de desarrollo Software buscan optimizar recursos en su etapa de producción, por esta razón el personal debe estar totalmente capacitado en cada una de las fases que intervienen en el proyecto, en la fase de requerimientos se aporta personal con trayectoria en desarrollo, de tal manera que permita descartar requerimientos funcionales no viables directamente con el cliente, en la fase de diseño se encuentra recurso humano con excelente formación en modelado y en la fase de desarrollo personal con conocimiento en técnicas y herramientas para realizar una producción eficiente y eficaz. ¿Dónde queda la fase de arquitectura?, un error frecuente es contratar un experto que construya la arquitectura del sistema y entregue los artefactos o modelos a la empresa para que se encargue del proceso. ¿Por qué no tener personal capacitado o con conocimientos claves que le permitan tomar una decisión en etapas tempranas?, con esto no se quiere decir que no sean necesarios los arquitectos de software, por el contrario, son de vital importancia, pero ¿por qué no multiplicarlos en la empresa para que exista un conocimiento explícito sobre

el tema en donde la decisión no la tome un experto, sino un equipo de trabajo sustentado en protocolos y procedimientos reales y probados en la práctica? Aprender de la experiencia es interesante, pero aprender de los errores de los demás y de las experiencias de los demás puede ser un factor clave de éxito en la industria de Software en Colombia y el mundo (Xiaoli et al., 2007).

Para el desarrollo del proyecto se realizaron las siguientes actividades: a) selección de proyectos de desarrollo software a partir de criterios que permitieron consolidar una muestra significativa para ser analizados; b) identificación de requisitos funcionales y no funcionales en los proyectos seleccionados para tipificar los requisitos comunes y definir el dominio de requerimientos, para esta actividad se revisó la documentación de la fase de diseño y se logró tener acceso a los proyectos para verificar las funcionalidades. c) Se realizó la selección de patrones arquitectónicos de referencia para asociarlos con los requerimientos del dominio de requerimientos definido y finalmente d) se validó el dominio de requerimientos para corroborar que los patrones arquitectónicos asociados, representan una opción pertinente para el requerimiento.

CONTEXTUALIZACIÓN

Arquitectura software

La -AS- se define como el diseño de mayor nivel de abstracción en la estructura de un sistema informático, se podría definir como el arte y la ciencia de diseñar software (Gutiérrez, 2010). Consiste en la agrupación de patrones y abstracciones que proveen de un marco de referencia que sirve como guía para la construcción de un software. En términos de un sistema o programa, la arquitectura de software se define como la estructura o estructuras del sistema, que comprende elementos de software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos. (Bass et al., 2012).

Patrones de arquitectura

La -AS-, según el estándar IEEE 1471-2000, se define como la organización fundamental de un sistema incorporando sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Un patrón de arquitectura busca solucionar problemas de adaptabilidad de requerimientos, rendimiento, modularidad y acoplamiento de los componentes de una aplicación. Los patrones de arquitectura, dentro o entre los niveles arquitectónicos, están relacionados con la interacción de los objetos. Los patrones de arquitectura, al igual que los patrones de diseño, representan llamadas entre objetos, decisiones y criterios arquitectónicos, así como la manera de empaquetado de funcionalidades de una aplicación. Sin embargo, los patrones de arquitectura representan un nivel más alto en el sistema.

Este tipo de patrones proveen un conjunto de subsistemas predefinidos, especifican las responsabilidades e incluyen reglas y guías para organizar las relaciones entre estos subsistemas. Algunos patrones de arquitectura son: a) Patrón Layers, logra descomponer la aplicación en un conjunto de capas jerárquicas y autónomas, las capas actúan de manera colaborativa, proveyendo servicios a la capa superior siguiente; b) Patrón Broker, se usa para dar organización a sistemas de tipo distribuido, que están compuestos por componentes débilmente acoplados y que interactúan entre sí invocando servicios remotos. Por esta razón es particularmente utilizado en arquitecturas orientadas a servicios SOA; y c) Patrón MVC - Modelo Vista Controlador, proporciona facilidad en la interacción entre el usuario final de una aplicación y la manera como se comunica con el servidor, MVC divide una aplicación en 3 áreas: procesamiento, salida y entrada, para esto, utiliza las abstracciones modelo - vista - controlador.

ANTECEDENTES

El uso de patrones arquitectónicos para mejorar el proceso de desarrollo de aplicaciones es abordado en (That et al., 2012), este trabajo presenta una propuesta para la formalización de técnicas con el fin de soportar el proceso de selección de patrones arquitectónicos y describe cómo integrar las decisiones arquitectónicas en un modelo de arquitectura. Autores como Sanchez et al. (2012) presentan un enfoque para analizar la aplicación de las tácticas de los patrones arquitectónicos. Estos últimos definen e ilustran el enfoque a partir de la especificación, análisis y verificación de patrones de arquitectura. Consiste en la caracterización de los principios de diseño de un patrón arquitectónico como las limitaciones expresadas en el idioma.

Otros autores como Yadav y Joshi (2010), se centran en la captura de los aspectos de la descripción de patrones y las propiedades de las interacciones entre componentes, incluyendo el comportamiento no determinista. El enfoque sirve de marco de referencia para las descripciones arquitectónicas precisas.

Existen algunas corrientes que se orientan hacia la reutilización de los patrones como es el caso de Cimpan y Couturier (2008) quienes detallan cómo lograr una mejora en los diseños de software utilizando los estilos arquitectónicos a partir de técnicas de reutilización. Como conclusión los autores expresan que la integración de estilos como medios para formalizar patrones arquitectónicos, complementaría una descripción de UML 2.0, aumentando el potencial de reutilización de tales patrones, sin embargo, existe un conjunto de capacidades principales que deben ser analizadas al seleccionar un estilo de definición de lenguaje para la formalización del patrón: a) capacidad de representar restricciones estructurales de las arquitecturas, b) capacidad de representar restricciones del comportamiento de las arquitecturas y c) posibilidad de utilizar plantillas o mecanismos de constructor, tales que las arquitecturas que sigan los estilos se pueden generar desde el estilo, tales mecanismos facilitarían la adaptación al patrón en un contexto específico.

Soluciones específicas a los problemas de arquitectura son descritos por Choppy et al. (2005) quienes proponen patrones de arquitectura de software que responden a necesidades puntuales en un proyecto de desarrollo software. Se muestra que estos patrones arquitectónicos reflejan exactamente las propiedades de los problemas y que se puede combinar de forma modular para resolver problemas de múltiples cuadros. También se proporcionan arquitecturas alternativas para hacer frente a las características específicas del sistema. Yau et al. (2005) ilustran la especificación de un lenguaje y herramientas para la especificación, análisis e implementación de patrones de arquitectura para sistemas software confiables. El lenguaje presentado es una extensión de SOL (Secure Operations Language), este lenguaje apoya la especificación del comportamiento y la composición e interacción entre los agentes. SOL extendido se puede aplicar en el diseño de los patrones arquitectónicos que implementen requisitos seguridad, tiempo real y tolerancia a fallos en una aplicación. El lenguaje SOL tiene una semántica bien definida para la composición, que permite el análisis automatizado de los diseños arquitectónicos y el impacto de las opciones de diseño de los requisitos de confiabilidad. Un caso de estudio se puede apreciar en la literatura (Erradi, 2012), el cual presenta el proceso de aplicación y evaluación de dos patrones de arquitectura para un sistema real en un contexto académico, el resultado refleja que la metodología aplicada por los autores es útil en un 90% de los estudiantes que participaron en el estudio. El manejo de ejemplos reales permite que los estudiantes se familiaricen eficientemente con los conceptos y apropien la fundamentación conceptual, al momento de aplicar patrones de arquitectura y diseño en sus proyectos.

Finalmente, Harrison et al. (2007) abordan la problemática relacionada con satisfacer las especificaciones funcionales del sistema a ser creado, sin descuidar el cumplimiento de sus necesidades no funcionales. Este trabajo se centra en las primeras etapas del proceso de arquitectura de software. Se estudia el problema de la partición de sistema con respecto a los dos requisitos funcionales y atributos de calidad. Existen métodos de diseño arquitectónico para acomodar el uso del patrón, pero no lo exploran en detalle. Se propone un patrón basado en el modelo que aprovecha las ventajas de los patrones, y encaja bien con los métodos existentes.

DEFINICIÓN DE LA TIPIFICACIÓN DEL DOMINIO DE REQUERIMIENTOS

La primera fase del trabajo presentado en este artículo contempló la selección de proyectos de desarrollo software, en (Guerrero et al., 2013) se establecen los criterios para evaluar y seleccionar procesos de desarrollo software de gran envergadura, éstos criterios permitieron establecer el tamaño de la muestra que sirvió de base para la presente investigación (68 proyectos). El trabajo realizado en (Guerrero et al., 2013) pudo ser reutilizado en su totalidad, debido a que los procesos analizados cumplían con la premisa de promover la integración de plataformas y ecosistemas digitales, el cual es un aspecto relevante en este estudio.

Un proyecto de desarrollo software formal cuenta con artefactos o productos en cada una de sus fases, la aplicación de criterios en la muestra inicial de proyectos permitió identificar aquellos que dentro de su proceso implementaron modelos de requisitos y modelos de calidad, bajo esta premisa, fue posible contar con los artefactos y entregables producidos como resultado del proceso, elementos que fueron utilizados como insumo en esta fase para la identificación de requisitos funcionales y no funcionales.

La estrategia para el análisis de cada proyecto se diseñó con tres actividades: a) revisión de los documentos de especificación de requisitos del software y del sistema: en esta actividad se utilizaron como insumo los documentos descritos anteriormente y se empleó la técnica de inspección documental, de tal forma que fue posible definir un listado con los requisitos de cada proyecto; b) ejecución de los proyectos software en un servidor local: la documentación de la fase de requerimientos y diseño facilitó el acceso a los requerimientos definidos, con el apoyo del director de cada proyecto fue posible ejecutar los proyectos software seleccionados en un entorno local, y de esta forma corroborar los requisitos funcionales que finalmente se implementaron; y c) recopilación de requisitos: los requisitos identificados se recopilaron en un instrumento

en formato Excel, que permitió la organización de los requisitos para lograr clasificarlos en funcionales y no funcionales, así como identificar los que se definieron para interactuar con otros sistemas.

Esta fase permitió identificar los requisitos transversales en todos los proyectos trabajados, a pesar que cada herramienta respondía a un proceso particular fue posible establecer los requisitos comunes. Otro aspecto clave fue identificar si estos requisitos comunes cumplían funcionalidades de interacción en un ecosistema digital o de integración de sistemas. En el caso de los requisitos no funcionales, todos los proyectos coincidieron en las mismas especificaciones, en el siguiente apartado serán presentados.

Tipificación del dominio de requerimientos

En el contexto de este proyecto, un dominio de requerimientos es definido como la agrupación de requisitos que tienen particularidades en común en varios sistemas o que se encuentran presentes en diversos sistemas. Ahora bien, con la identificación de los requisitos en cada proyecto, se logró realizar la tipificación de aquellos comunes a los sesenta y ocho proyectos de la muestra, esta tipificación permitió conformar el dominio de requerimientos. En la Tabla 1 se presentan la tipificación de requisitos funcionales y en la Tabla 2 no funcionales que conforman el dominio de requerimientos comunes, identificados en los proyectos analizados.

Tabla 1: Requisitos funcionales comunes

ID	*Requisitos comunes*
RF1	Autenticación
RF2	Autorización
RF4	Asignación de funcionalidades
RF5	Envío y recepción de archivos
RF6	Operaciones CRUD (Create, Read, Update, Delete)
RF7	Importación de data desde archivos externos
RF8	Exportación de datos a archivos externos
RF9	Envío y recepción de datos de acuerdo a un modelo definido
RF10	Interconexión entre componentes - servicios
RF11	Interconexión entre componentes - puertos
RF12	Abstracción de sistemas de bases de datos relacionales

Tabla 2: Requisitos no funcionales comunes

ID	*Requisitos comunes*
RNF1	Disponibilidad
RNF2	Tiempos de respuesta rápidos (alto rendimiento)
RNF3	Integridad de la información
RNF4	Seguridad
RNF5	Mantenibilidad
RNF6	Interoperabilidad

Patrones arquitectónicos para el dominio de requerimientos

La arquitectura software se convierte en parte primordial en el proceso de desarrollo del software, y es por esta razón que se debe contemplar la arquitectura de software como una fase inamovible, que requiere de personal capacitado y con experticia en el área (Jain y Kircher, 2007). Un patrón arquitectónico surge cuando es una solución que ha sido útil en el tiempo y en diferentes contextos, por lo tanto, es posible encontrarlo documentado (Gamma et al., 1994).

Los patrones de arquitectura proporcionan estrategias de empaquetado para resolver algunos de los problemas que enfrenta un sistema. Un patrón arquitectónico delinea los tipos de elementos y las formas de interacción utilizadas para solucionar un problema (Bass et al., 2012). La experiencia del equipo de trabajo permite inferir que el interés por los patrones de arquitectura se remonta alrededor de 15 años, durante este tiempo el concepto ha madurado y han ido apareciendo los patrones de arquitectura más utilizados, debido a que por definición los patrones se definen en la práctica, es decir, que se descubren, fortalecen y formalizan con el tiempo, resulta una tarea compleja tener un catálogo definitivo de los patrones de arquitectura.

La revisión bibliográfica realizada en el desarrollo de este proyecto, permitió identificar tres principales fuentes de información que abordan la temática de los patrones de arquitectura y describen conjuntos de patrones de arquitectura. En la Tabla 3 se presentan los referentes bibliográficos.

Tabla 3: Catálogos de Patrones de Arquitectura

Nombre del Libro	Autor	Descripción
Software Architecture in practice (Bass et al., 2012)	Bass, Leon Clement, Paul Rick, Kazman	Es la referencia más reciente con un compendio de tácticas y patrones de arquitectura. Ofrece un recorrido por el universo patrones de arquitectura y describen los 11 patrones, que de acuerdo a los autores son los más empleados en el diseño de arquitecturas software.
Patterns of Enterprise Application Architecture (Fowler y Rice, 2002)	Martin Fowler David Rice	Contiene alrededor de 40 patrones para apoyar la etapa de arquitectura, durante el proceso de desarrollo software de tipo empresarial. Los patrones de este catálogo se agrupan en las siguientes categorías: • Layering • Organización de la lógica del dominio • Mapping to Relational Databases • Presentación Web • Concurrencia • Estado de sesión • Estrategias de Distribución
Pattern Oriented Software Architecture (Frank et al., 1996)	Frank Buschmann Regine Meunier Hans Rohnert Peter Sommerlad Michael Stal	Es un catálogo de Patrones de Arquitectura, actualmente en su última edición. Contiene 8 patrones agrupados en las siguientes categorías, según su enfoque: • From Mud to Structure • Sistemas Distribuidos • Sistemas Interactivos • Sistemas Adaptables

Después de analizar las diferentes fuentes primarias de información, se procedió a seleccionar la de mayor pertinencia para definirla como referencia, para lo cual se determinó como principal criterio la cercanía al tiempo presente como referente de una fuente actual, la razón de este criterio está dada por la misma naturaleza evolutiva de la tecnología y el desarrollo de software, las técnicas y patrones utilizados hace más de 10 años han evolucionado, es probable que conserven los mismos nombres, pero han evolucionado para adaptarse a las necesidades del contexto. De acuerdo a lo expuesto anteriormente, se seleccionó como bibliografía de referencia: Software Architecture in Practice de (Bass et al., 2012). Además de ser un documento recientemente actualizado, se encuentra avalado por el SEI - Software Engineering Institute, lo cual lo convierte en una referencia ampliamente confiable. A continuación, en la Tabla 4 se presentan los patrones de arquitectura de referencia para este estudio.

Tabla 4: Patrones de arquitectura (Bass et al., 2012)

Categoría	*Patrón*
Patrones de módulos	Patrón de capas
Patrones de componentes y conectores	Patrón bróker
	Patrón modelo vista controlador
	Patrón tuberías y filtros
	Patrón cliente servidor
	Patrón punto a punto
	Patrón Arquitectura Orientada a Servicios
	Patrón Publish-Suscribe
	Patrón Shared Data
Patrones de asignación	Patrón Map-reduce
	Patrón multi niveles

Patrones para requerimientos

En este punto del desarrollo del proyecto, se logró consolidar el dominio de requerimientos conformado por los requerimientos funcionales y no funcionales comunes que se encontraron en los proyectos analizados y los patrones de referencia que serán tomados como línea de base. El principal objetivo en esta actividad consistió en analizar cada uno de los requisitos y los patrones arquitectónicos de referencia, se realizó un proceso de verificación del código de implementación del requisito y a partir de la definición del patrón fue posible establecer una relación directa entre el requerimiento implementado con la definición conceptual, este proceso permitió encontrar equivalencias y similitudes, derivando en su posible uso.

En este punto es importante aclarar que en la documentación de los proyectos no se mencionaba el uso de patrones arquitectónicos, los desarrolladores aplicaron los conceptos del patrón arquitectónico sin conocerlo. Esto es común encontrarlo en proyectos que trabajan con estándares y buenas prácticas, las buenas prácticas redundan en sistemas desarrollados de forma organizada. En la Tabla 5 se presentan los requisitos funcionales del dominio de requerimientos y el patrón de arquitectura relacionado.

Tabla 5: Patrones de arquitectura usados en requisitos funcionales

ID	*Requisitos*	*Patrón arquitectónico*
RF1	Autenticación	Arquitectura Orientada a Servicios
RF2	Autorización	Arquitectura Orientada a Servicios
RF4	Asignación de funcionalidades	MVC
RF5	Envío y recepción de archivos	Arquitectura Orientada a Servicios
RF6	Operaciones CRUD (Create, Read, Update, Delete)	Capas, MVC
RF7	Importación de data desde archivos externos	Tuberías y filtros
RF8	Exportación de datos a archivos externos	Tuberías y filtros
RF9	Envío y recepción de datos de acuerdo a un modelo definido	Arquitectura Orientada a Servicios
RF10	Interconexión entre componentes - servicios	Arquitectura Orientada a Servicios
RF11	Interconexión entre componentes - puertos	Punto a punto
RF12	Abstracción de sistemas de bases de datos relacionales	Capas

En la Tabla 6 se presentan los requisitos no funcionales del dominio de requerimientos y el patrón de arquitectura relacionado. Como se observa en la tabla, cuatro requisitos no tienen relacionado un patrón arquitectónico, esto debido a que al momento de realizar la verificación del código no se encontraron características que permitieran relacionar la solución aplicada para cumplir con el requerimiento con un patrón específico.

Tabla 6: Patrones de arquitectura usados en requisitos no funcionales

ID	*Requisitos*	*Aplicación de patrón arquitectónico*	*Patrón arquitectónico*
RNF1	Disponibilidad	No	-
RNF2	Tiempos de respuesta rápidos (alto rendimiento)	No	-
RNF3	Integridad de la información	No	-
RNF4	Seguridad	No	-
RNF5	Mantenibilidad	Si	Capas, MVC
RNF6	Interoperabilidad	Si	Arquitectura Orientada a Servicios

En este punto se puede concluir que para proyectos Web que promueven la integración de plataformas o ecosistemas digitales, existe un dominio de requerimientos que agrupa los requisitos comunes. Adicionalmente, a partir de la asociación que se realizó de los requisitos con los patrones de arquitectura, se logró establecer una relación directa con los patrones que son los más indicados para cada requerimiento.

Validación de la tipificación del dominio de requerimientos

La validación se realizó en una institución de educación superior del sector público, denominada Unidades Tecnológicas de Santander -UTS-, en el marco del proyecto de investigación: *Ecosistema Digital Académico Caso Aplicado: Unidades Tecnológicas de Santander -Fase inicial-*. Este proyecto tuvo como objetivo diseñar la arquitectura software y la estrategia tecnológica que permitiera la implantación de un ecosistema digital de carácter académico para las UTS, buscando la optimización de procesos y recursos. En el marco del proyecto mencionado anteriormente, se seleccionó el macro-proceso denominado Evaluación Docente, teniendo en cuenta que es un proceso de vital importancia para los principales actores de la institución: estudiantes, docentes y administrativos, es decir que afecta al 100% de los involucrados en los procesos académicos de las UTS, que suman en promedio 22.000 personas. El proceso se realiza directamente desde la intranet institucional que es accesible desde la Web.

La difusión de los resultados de la evaluación docente, solo está habilitado para los actores con perfil directivo como apoyo a la toma de decisiones, sin embargo, con el fin que esta actividad sea transparente, se propuso el desarrollo de una aplicación para dispositivos móviles con sistema operativo Android, para que los docentes, por medio de la autenticación y autorización, tuvieran acceso a sus resultados personales. A continuación, se relacionan los requisitos definidos para el prototipo. a) Autenticación de usuarios con el usuario y la contraseña utilizados para ingresar a la intranet. b) Autorización para acceder al componente de la evaluación para el perfil docente; c) carga de la información personal del usuario y la foto de la hoja de vida alojada en la intranet; d) carga de los resultados de las evaluaciones docentes en las cuales haya participado el usuario.

Los requisitos presentados anteriormente se implementaron en una aplicación móvil para dispositivos Android y se utilizó la tipificación de dominios de requerimientos y patrones arquitectónicos para validar la consistencia de los resultados de este estudio, para realizar la validación se realizó un proceso de asociación entre los requisitos del prototipo contra los requisitos del dominio de requerimientos. En la Tabla 7 se presentan los requisitos del prototipo y los requisitos del dominio de requerimientos con el patrón de arquitectura correspondiente. La aplicación móvil desarrollada se encuentra publicada en la tienda Google Play y a la fecha está en el rango de instalaciones de 1.000 - 5.000.

Tabla 7: Validación del dominio de requerimientos

ID	*Requisitos*	*Requisito del dominio de requerimientos*	*Patrón de arquitectura*
1	Autenticación de usuarios con el usuario y la contraseña utilizados para ingresar a la intranet.	RF1 - Autenticación	Arquitectura Orientada a Servicios
2	Autorización para acceder al componente de la evaluación para el perfil docente.	RF2 - Autorización	Arquitectura Orientada a Servicios
3	Cargar la información personal del usuario y la foto de la hoja de vida alojada en la intranet.	RF5 - Envío y recepción de archivos	Arquitectura Orientada a Servicios
4	Carga de los resultados de las evaluaciones docentes en las cuales haya participado el usuario.	RF9 - Envío y recepción de datos de acuerdo a un modelo definido	Arquitectura Orientada a Servicios

CONCLUSIONES

El resultado de este trabajo permitió ampliar el panorama entregado por los trabajos relacionados, es un aporte a las diferentes metodologías, tácticas y estrategias para el diseño de una arquitectura software, puesto que parte del principio de los requerimientos funcionales de un proyecto software y define puntualmente cuál patrón de arquitectura utilizar a partir de un requerimiento específico. Por tal razón, es una herramienta adicional que podrá ser tenida en cuenta en la fase de arquitectura software de un proyecto.

Cuando en un proyecto de desarrollo software se aplican estándares y buenas prácticas, la estructura de código es legible gracias a su organización. Los proyectos analizados utilizaron como línea de base el patrón MVC al iniciar los desarrollos sobre un marco de trabajo, la estructura de MVC facilita la navegabilidad entre cada una de las capas del proyecto para el análisis del código.

La identificación de patrones de arquitectura en los proyectos seleccionados requirió el mayor esfuerzo en este proyecto, debido a que su implementación no se encontraba documentada como con los patrones GOF, es posible inferir que no se emplearon técnicas para seleccionar patrones de arquitectura, sin

embargo, dada la madurez del equipo perteneciente al proyecto el diseño coincidió con las características de los patrones que se lograron relacionar.

No fue posible identificar patrones arquitectónicos para todos los requisitos no funcionales comunes en los proyectos analizados, el proceso de identificación presentó un nivel de complejidad mayor. Es posible inferir que la solución utilizada no fue un patrón de arquitectura, una posible solución utilizadas son las tácticas para el modelado de atributos de calidad.

La clasificación de patrones de arquitectura que se realizó para la tipificación de requerimientos, es viable para ser aplicada en el contexto real, en equipos de desarrollo, pero, se requiere una estrategia de sensibilización en el uso de patrones de arquitectura. Aunque actualmente la fase de diseño de arquitectura está siendo contemplada en un proceso de desarrollo, no implica que se estén aplicando buenas prácticas como la implementación de patrones.

Los proyectos que se analizaron cumplieron los criterios de selección para garantizar proyectos de mayor calidad, el uso de un lenguaje de modelado como UML representó un apoyo para la identificación de patrones. Un caso opuesto es analizar el grupo de proyectos que no cumplieron el requisito de diseños UML, para este caso la estrategia debe ser diferente debido a que estaría totalmente orientada al análisis del código de la aplicación y la estructura interna, el resultado podría ser comparado con el análisis realizado a los proyectos que si cumplieron todos los criterios de selección.

El prototipo que permitió realizar la validación del dominio de requerimientos, responde a un proceso puntual que hace parte de un ecosistema digital académico, la asociación de los requisitos de la aplicación con el dominio de requerimientos permite concluir que, para una aplicación, en este caso móvil, que demanda información de otro sistema para integrarse en el ecosistema digital, el patrón más indicado es el de arquitectura orientada a servicios.

REFERENCIAS

Ampatzoglou, A., Chatzigeorgiou, A., Charalampidou, S., Avgeriou, P. *The Effect of GoF Design Patterns on Stability: A Case Study*. doi: 10.1109/TSE.2015.2414917. IEEE Transactions on Software Engineering (en línea), 41(8), 781-802 (2015)

Bass, L., Clements, P., y Kazman, R. *Software Architecture in Practice* (3 edición). 203-249. Pearson Educación. Inc. Massachusetts, USA. (2012)

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., y Stal, M. *Pattern-Oriented Software Architecture*-25-193. John WIley & Sons. New York, USA (1996)

Choppy, C., Hatebur, D., y Heisel, M. *Architectural patterns for problem frames*. doi: 10.1049/ip-sen:20045061. IEEE Proceedings – Software (en línea), 152(4), 198-208 (2005)

Cimpan, S., y Couturier, V. *Can Styles Improve Architectural Pattern Reuse?* doi: 10.1109/WICSA.2008.38. Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008) (en línea), 263-266 (2008)

Dabous, F. T., Rabhi, F. A., y Yu, H. *Using software architectures and design patterns for developing distributed applications*. doi: 10.1109/ASWEC.2004.1290482. Australian Software Engineering Conference. Proceedings. 290-299 (2004)

Erradi, A. *Applying and evaluating architectural patterns on a stock trading case study*. doi: 10.1109/ICITeS.2012.6216670. International Conference on Information Technology and e-Services (en línea). 1-6 (2012)

Fowler, M., y Rice, D. *Patterns of Enterprise Application Architecture*. 17-103 Addison-Wesley Professional. Massachusetts, USA (2002)

Gamma, E., Helm, R., Johnson, R., y Vlisides, J. *Design Patterns Elements of Reusable Object Oriented Software*. 1-358. Addison Wesley Longman Inc. USA (1994)

Guerrero, C. A., Suárez, J. M., y Gutiérrez, L. E. *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web*. doi: 10.4067/S0718-07642013000300012. Información Tecnológica (en línea), 24(3), 103-114 (2013)

Gutiérrez, L. E., *Arquitectura Software, Investigación Aplicada a la Construcción de Marcos de Trabajo*. 10-48, (Sic) Editorial Ltda., Bucaramanga, Colombia. (2010)

Harrison, N., Avgeriou, P. *Pattern-Driven Architectural Partitioning: Balancing Functional and Non-functional Requirements*. doi: 10.1109/ICDT.2007.65. Second International Conference on Digital Telecommunications (ICDT'07) (en línea) 21-26 (2007)

IEEE y ACM. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. doi: 10.1145/2534860. ACM, Inc. (en línea). USA (2013)

Klein, J. *What Makes an Architect Successful?* doi: 10.1109/MS.2016.9. IEEE Software (en línea), 33(1), 20-22 (2016)

Lago, P., y Vliet, H. Van. *Teaching a Course on Software Architecture*. doi: 10.1109/CSEET.2005.33. Conference on Software Engineering Education & Training (CSEET'05) (en línea), 35-42 (2005)

Patidar, A., y Suman, U. *A survey on software architecture evaluation methods.* 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 967–972. (2015)

Sanchez, A., Aguiar, A., Barbosa, L. S., y Riesco, D. *Analysing Tactics in Architectural Patterns*. doi: 10.1109/SEW.2012.10. 35th Annual IEEE Software Engineering Workshop (en línea), 32-41 (2012)

That, M. T. T., Sadou, S., y Oquendo, F. *Using Architectural Patterns to Define Architectural Decisions*. doi: 10.1109/WICSA-ECSA.212.28. Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (en línea), 196-200 (2012)

Xiaoli, L., Guoqing, W., Min, J., Min, Y., y Weiming, W. *Software architecture for a pattern based Question Answering system*. doi: 10.1109/SERA.2007.120. 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007) (en línea), 331-336 (2007)

Yadav, D. K., y Joshi, R. K. *Capturing interactions in architectural patterns*. doi: 10.1109/IADCC.2010.5422893. 2nd International Advance Computing Conference (IACC) (en línea), 443-448 (2010)

Yau, S., Mukhopadhyay, S., y Bharadwaj, R. *Specification, Analysis and Implementation of Architectural Patterns for Dependable Software Systems*. IEEE. doi: 10.1109/WORDS.2005.5. 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (en línea), 197-204 (2005)