REVISTA TIA

Tecnología, Investigación y Academia -Red Avanzada – RITA
Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología

Especificaciòn de la arquitectura de software

Software architecture specification

Luz Amanda Quilindo [1], Juan Sebastian Vega [2]

Resumen:

En la siguiente investigación se evidenciarán los componentes que forman parte de una arquitectura de software y su relación a través de los patrones arquitecturales; día a día son más los avances tecnológicos que encontramos en la interacción humana, el crecimiento de plataformas móviles y web expandidos en el mundo ha cambiado la forma de pensar y diseñar una infraestructura requerida para funcionar estos procesos. Tareas cada vez más complejas para el software, tiempos cada vez más cortos de entrega y procesamiento de datos, seguridad en las diferentes transacciones y una arquitectura sólida para soportar la magnitud de una idea en crecimiento, son solo algunos de los detalles que los patrones arquitecturales pretender organizar y esquematizar, este artículo brinda una vista rápida a cada uno de los componentes, sus interpretaciones y contexto dentro de la arquitectura de software que trabajamos en la actualidad.

***Palabras clave**: Arquitectura de Software, Patrones Arquitecturales, Sistemas.*

[1.] Especialista en procesos y calidad, Universidad EAN, Sistemas Inteligentes en Red, 0000-0002-1131-5070, luzamanda@gmail.com, Colombia

[2. 2] Ingeniería de Sistemas, Universidad Central, ScotiaBank, 0000-0002-5507-8054, juan_sebas9419@hotmail.com, Colombia

Vol. 11 No. 2
Fecha de aceptación: marzo 2023
ISSN: 2344 8288
Artículo corto
Fecha de recepción: Enero -2023.

Abstract

In the following investigation you will find evidence of the different components that are part of a software architecture and their relationship through architectural patterns; day by day there are more technological advances that we find in human interaction, the growth of different mobile and web platforms that expand throughout the world, has changed the way of thinking and designing an entire infrastructure required for the operation of these processes. More and more complex tasks for the software, ever shorter delivery and data processing times, security in the different transactions and a solid architecture to support the magnitude of a growing idea, are just some of the details that the architectural patterns intending to organize and outline, this article provides a quick view of each of the components, their interpretations and context within the software architecture that we work with today.

Keywords: *Software Architecture, Architectural Patterns, Systems.*

I. INTRODUCCIÓN

Los patrones de software han ido cambiando con el paso del tiempo dentro del ambiente tecnológico en el que nos encontramos, cada uno de estos se ha ido ajustando cada vez más a las nuevas tendencias en arquitectura de software, soportando diferentes tipos de integraciones y estructuras empresariales. Los patrones de software han ido transformando la forma en la que diseñamos, implementamos y pensamos acerca de un problema, estos patrones nos proveen de un vocabulario propio de la arquitectura y estructuras sólidas para la construcción del software. Pequeñas piezas presentadas como guías para mejorar la comunicación, estructura y consistencia del proyecto.

Los patrones son creados a partir de la experiencia colectiva de diseñadores e ingenieros de software. Estos expertos ya tienen soluciones a muchos de los problemas recurrentes en el diseño de software. Y es así, como se hace importante plasmar o concretar en un modelo estas soluciones, eliminando así la ambigüedad y haciendo más fácil la comunicación, cuando nuevamente se quiera recurrir a dicha solución.

En la misma medida que ha ido creciendo el conocimiento, la experiencia y la aplicabilidad de los patrones, han crecido también, las diferentes variaciones de cada uno de estos patrones, sus propiedades, sus relaciones con otras tecnologías y con otros patrones [17], Son pocos los nuevos patrones concretos que se han documentado desde mediados de los 90's [1][2].

Pero los patrones no se comportan como islas independientes y pueden trabajar en conjunto con otros patrones para dar soluciones a otros problemas, de estas combinaciones nacen conceptos como: patrones stand-alone, patrones complementos, patrones compuestos, patrones historias, patrones secuencia y el lenguaje de patrones.

Cada uno de estos patrones conlleva a una exploración misma del software, la caracterización de las técnicas y tecnologías aplicadas y el desarrollo de conceptos viables para su implementación.

II. MARCO TEÓRICO

Según Kruchten [8] las arquitecturas son un proceso creativo y estas van muy ligadas a lo que los arquitectos involucrados consideren convenientes basados en las siguientes tres fuentes:

- **Método:** Puede ser visto como una manera concisa y documentada, mediante el cual la arquitectura es derivada desde los requerimientos del sistema y las restricciones tecnológicas.
- **Intuición:** Habilidad de concebir sin razonamiento consciente.
- **Reutilización:** La mayoría de los elementos de una arquitectura son adoptados de otras arquitecturas, ya que el arquitecto puede estar familiarizado con la problemática o con alguna arquitectura encontrada en la literatura técnica.

Lo anterior se puede resumir en el siguiente diagrama [15]

Ilustración 1: Diseño de una Arquitectura de Software [15]

Como se observa en el diagrama después de reunir los requerimientos del sistema a implementar, el primer paso es indagar sobre los patrones de referencia para escoger una estrategia general basados en estos patrones ya probados.

Los patrones más conocidos, aprobados y soportados son los presentados en libro "*Pattern Oriented Software Architecture*" [2] conocidos como los patrones POSA y también los del libro "*Pattern of Enterprise Application Architecture"* conocidos como PEAA.

Con el fin de acelerar el proceso de construcción de software es importante entender que los patrones se dividen en varios tipos, los principalmente son:

Patrones de arquitectura: Afectan la estructura global del sistema. Representan el nivel más alto de abstracción. La manera más clásica de clasificar estos patrones es como indica el libro "*Pattern Oriented Software Architecture POSA*"- 1996. Existe una forma más práctica de ver estos patrones clasificados, esta idea es presentada y explicada en el libro "*Pattern of Enterprise Application Architecture PEAA*" - Martin Fowler 2003.

Patrones de diseño: Afecta a subsistemas o componentes del sistema global y sus relaciones. Los más aceptados son los encontrados en el libro "*Design Pattern GOF*" donde se presentan 23 patrones. Pero también hay otras alternativas de clasificación como las presentadas en el libro *J2EE PATTERNS Best Practices and Design Strategies* donde realiza una división de 5 capas de arquitectura y 15 patrones de diseño o las presentadas también en "Pattern Oriented Software Architecture POSA"-1996.

Patrones de lenguajes: Son específicos de un lenguaje de programación y describen como implementar ciertos aspectos de un problema utilizando las características específicas de dicho lenguaje, el diseño de una arquitectura de software en relación con una solución puede estar compuesto por uno o varios de estos patrones como se evidencia en el siguiente diagrama:

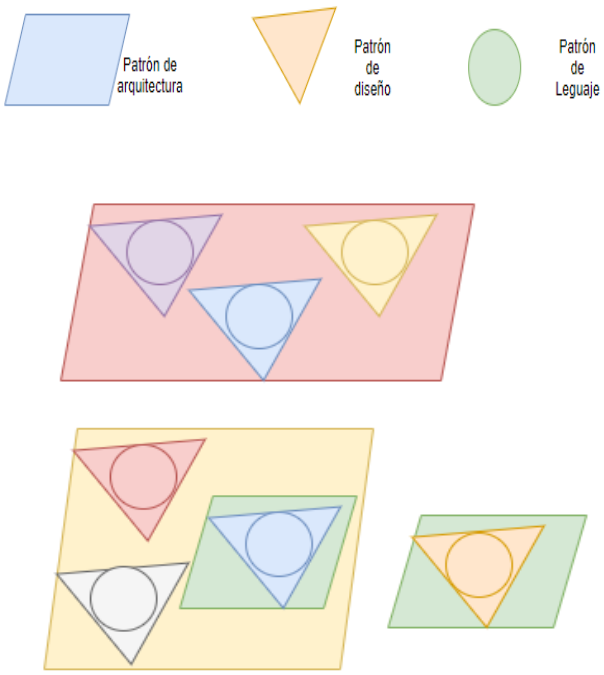

Ilustración 2: Ejemplos de sistemas de software con diferentes patrones.

Los patrones POSA, por ejemplo, divide los patrones arquitectónicos en cuatro categorías [2]:

- **Del lodazal a la estructura**: Los patrones de esta categoría ayudan a pasar un mar de componentes a tener una estructura organizada colaborando entre sí. (Patrones: capas, tuberías y filtros, Pizarra)**.**
- **Sistemas distribuidos:** Comunicación entre aplicaciones, validando el grado de conocimiento mutuo para poder intercambiar mensajes. (Patrones*: Broker*)**.**
- **Sistemas interactivos:** Cuyo objetivo es mejorar la usabilidad de una aplicación. Manteniendo el núcleo independiente de la interfaz de usuario. Estos se suelen usar en los subsistemas (Patrones: modelo-vista-controlador, Presentación-abstracción-control)
- **Sistemas adaptables:** Apoya la extensión de aplicaciones y a su adaptación a la tecnología en evolución (Patrones: *Microkernel y Reflection*).

Se pueden dividir los patrones de diseño en cinco categorías:

- **Descomposición estructural:** Ayuda a una apropiada descomposición de un subsistema, es decir a pasar de un componente complejo a diferentes partes cooperativas. (Patrón: Todo - Parte)**.**
- **Organización de trabajo:** Define como colaborar entre diferentes componentes para resolver un problema. (Patrón: Maestro- esclavo)**.**
- **Control de acceso:** Controlan el acceso de los componentes, prefieren la comunicación con un representante del componente que con el componente mismo. (Patrón: Proxy)**.**
- **Administración:** Define como manejar elementos similares como objetos, servicios o componentes. (Patrón: *Command Processor, View Handler*)**.**
- **Comunicación:** Ayuda a organizar la comunicación entre componentes. (*Patrón: Forward-Receiver, Client-Dispatcher-Server, Publisher-Subscriber).*

En relación con la definición de los componentes o elementos de una arquitectura de software, es necesario especificar cuáles son los componentes que harán parte de la aplicación, teniendo en cuenta que la arquitectura de referencia define los patrones de comunicación en general para los componentes. Para esto es importante entender cómo los patrones se relacionan con otros patrones [17], entendemos estas relaciones entre patrones de la siguiente manera:

- **Patrones Stand-Alone:** Son los más conocidos y se ha venido mencionando en este artículo como patrones de referencia [1][2] están documentados en las tres perspectivas. Concepto - Implementación- Documentación.
- **Patrones Complementos:** Donde un patrón provee el ingrediente que le hace falta a otro para hacer el diseño más completo y balanceado. De esta manera al implementarlos de forma conjunta pueden cooperar uno con el otro haciendo la solución más completa.
- **Patrones Compuestos:** Surge de la intersección de dos patrones que hace que el diseño se vaya hacia una sola decisión. Para llegar a sacar lo mejor de cada uno de estos patrones se debe llegar a un punto de granularidad importante.
- **Patrones Históricos:** Ofrece una presentación de cómo los patrones deben ser implementados para transformar una determinada situación.
- **Patrones Secuencia:** Generaliza los patrones históricos para que puedan ser implementados en diferentes lugares dependiendo el contexto.

Christopher Alexander [26] en su libro de Arquitectura de Edificios, describe una metodología que facilita este análisis de relaciones entre patrones y entre las diferentes variables del contexto que resultan convenientes para el desarrollo de software con patrones arquitecturales, poder relacionarlos e implementarlos de la mejor manera generará como resultado una estructura fundamentada y muy bien organizada del proyecto a desarrollar.

Las diferentes vistas son entregadas de forma que logremos analizar los componentes que hacen parte de ella, separando aquellos que hacen parte de la aplicación y aquellos que hacen parte del área técnica. En el proceso de formar los diferentes puntos de vista de un sistema, se puede relacionar con otros conjuntos de vistas, para describir las dependencias entre ambos sistemas y llegar a plasmar una infraestructura de forma general de la solución diseñada.

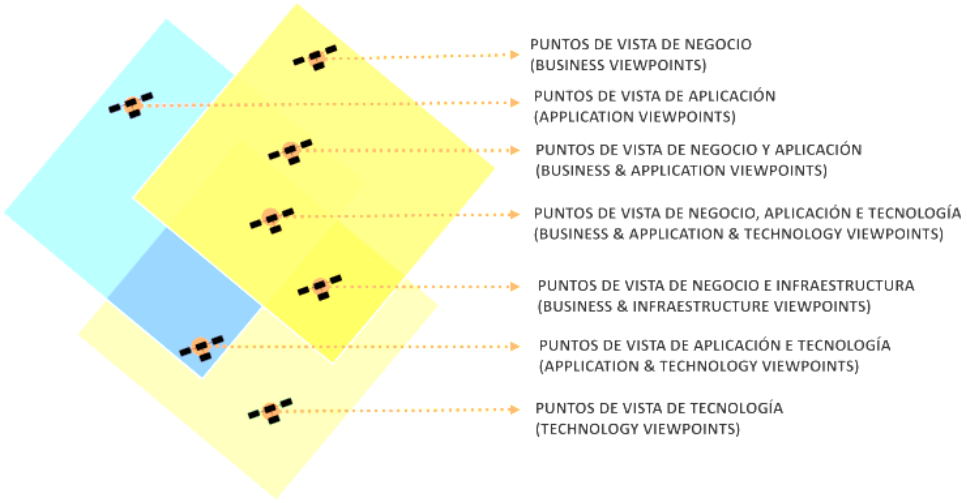

Ilustración 3: Puntos de Vista

Existen detalles que se debe tomar en cuenta al momento de diseñar utilizando un modelamiento de puntos de vista, como la importancia de entender la diferencia entre dominios y el propósito de cada punto de vista. Los puntos de vista deben estar descritos en un lenguaje de modelado estándar que no de opción a malentendidos; no es necesario que todos los puntos de vista se describan en el mismo lenguaje de modelado, pero si es necesario que exista una trazabilidad entre los lenguajes utilizados.

Los patrones son un medio para documentar las arquitecturas de software. Esta documentación permite rastrear con el pasar del tiempo determinados cambios y flujos alternos, porque algunas elecciones se hicieron y porque otras elecciones fueron rechazadas. De esta manera el mantenimiento del flujo propuesto puede ser llevado a cabo de una mejor manera.

La documentación de las arquitecturas basadas en patrones ayudará a resolver conflictos y a esclarecer dudas con el pasar del tiempo y cuando sea necesario hacer ajustes o modificaciones, mejoras de rendimiento, o profundizar en los patrones de seguridad [22] y en otros sistemas. Detallando estos patrones [23][24][25] se pueden crear y entender todo tipo de sistema.

La documentación de los patrones es supremamente importante en el diseño de una arquitectura y en el entendimiento mismo de los patrones. Desafortunadamente muchos arquitectos o desarrolladores tienden a pensar que los patrones son asuntos fijos como planos o una configuración de clases muy específicas.

Se sugiere que la presentación de un patrón puede darse de la siguiente manera:

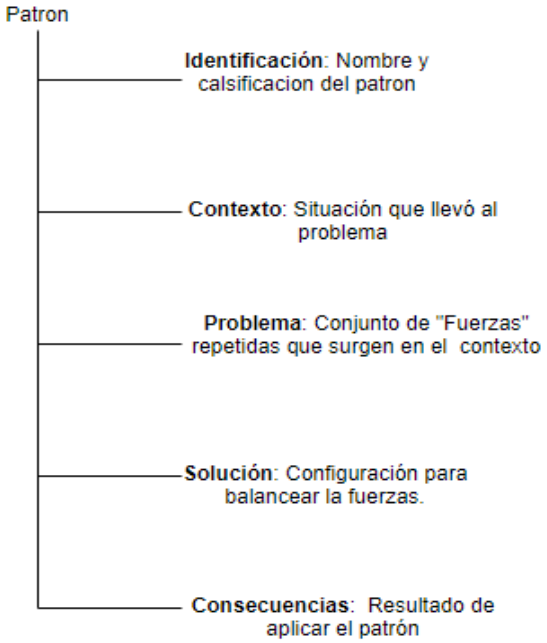

Ilustración 4: Presentación de los patrones

- Contexto: Juega un importante rol en el patrón, ya que el especificarse no se caerá en el error de usarse inadecuadamente. Si el contexto es demasiado amplio se caerá en el error de pensar en un patrón todo terreno, pero maestro de ninguno.
- Problema: Definición de las fuerzas que se repiten en un contexto y que puede llevar a que el sistema trabaje adecuadamente o con poco rendimiento sino se logra manejar adecuadamente.
- Solución: A menudo la solución suele acompañarse con un pedazo de código, pero una imagen dice más que mil palabras y ayuda a comunicar la esencia de cómo el patrón soluciona el problema.
    - Consecuencias: Ventajas que trae la implementación de este patrón y como facilita y balancea las fuerzas del problema y en caso de tener alguna consecuencia negativa mencionarla para que se pueda determinar el peso de la misma en determinado contexto.

De igual manera entre más claridad presente en la documentación esta le servirá más adelante, podrían incluirse otros elementos como estructura, dinámicas, implementaciones posibles, usuarios del patrón, entre otros. La habilidad del arquitecto en expresar y plasmar el diseño jugará un gran papel.

III. CONCLUSIONES

Algunos desarrolladores y arquitectos sin experiencia pensaban que los patrones ayudarían a formular las arquitecturas de un sistema complejo, como siguiendo una receta, mediante la aplicación mecánica de pasos. La experiencia ha concientizado que la integración de estos patrones a un sistema combina las habilidades humanas de capturar propiedades, generalizar, concretizar, sumado a técnicas y herramientas e incluso poder llevar todo este conocimiento a un nuevo nivel para explorar nuevas ideas y combinaciones.

Referencias

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: PatternOriented Software Architecture, Volume 1: A System of Patterns, John Wiley & Sons, 1996

[3] Enterprise Development Reference architecture, Microsoft

[4] J2EE PATTERNS Best Practices and Design Strategies

[5] R. C. advance, «RJ Code Advance,» 25 junio 2019. [En línea]. Available: https://rjcodeadvance.com/patrones-de-software-que-es-patron-de-arquitectura-parte-3/. [Último acceso: 05 2021].

[6] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, Second Edition. Addison-Wesley, 2003.

[7] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, Robert Nord, and Judith Stafford. Documenting Software Architectures: Views and Beyond (2nd Edition). Addison-Wesley, 2010

[8] P. Kruchten. Mommy, where do software architecture come from? 1st International Workshop on Architectures for Software Systems (IWASS1), pages 198–205, 1995.

[9] R. N. Taylor, N. Medvidovic, and E. M. Dashofy. Software Architecture: Foundations, Theory, and Practice. Wiley, 2010.

[10] Ch. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America. A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software, 80(1):106–126, 2007.

[11] T. B. Bollinger and S. L. Pfleeger. The economics of reuse: Issues and alternatives. In GA Atlanta, editor, Proceedings of the Eighth Annual National Conference on Ada Technology, pages 436–447, 1990.

[12] D. Gentner. Similarity and analogical reasoning. chapter The Mechanisms of Analogical Learning, pages 197–241. Cambridge University Press, 1989.

[13] H. Gust, U. Krumnack, K. U. Kuhnberger, and A. Schwering. Analogical¨ reasoning: A core of cognition. Zeitschrift fur Künstliche Intelligenz (KI), ¨ Themenheft KI und Kognition, (1):8–12, 2008

[14] M. C. Carignano, «Representación y razonamiento sobre las decisiones de diseño de arquitectura de software.,» Facultad Regional Santa Fe, Universidad Tecnológica Nacional, 2015.

[15] F. B. A. MS.c, «Modelado y Diseño de Arquitectura de Software,» Universidad Javeriana, Cali.

[16] F. J. G. Peñalvo, «Patrones,» Universidad de Salamanca..

[17] Pattern Oriented Software Architecture POSA"-1996 Volimen 5

[18] W. T.-B. ,. J. I. R.-P. Victor Hugo Jimenez-Torres, «Lenguajes de Patrones de Arquitectura de Software: Una Aproximación al estado del arte,» *Scientia et Technica*, vol. 19, nº 4, p. 371, 2014.

[19] J.O. Coplien, N.B. Harrison: Organizational Patterns of Agile Software, Development, Prentice Hall, 2004

[20] R. Porter, J.O. Coplien, T. Winn: Sequences as a Basis for Pattern Language Composition, Science of Computer Programming, Elsevier, 2005

[21]. K. Henney: Context Encapsulation – Three Stories, A Language, and Some Sequences, Proceedings of the Tenth European Conference on Pattern Languages of Programming (EuroPloP 2005), Irsee, Universitätsverlag Konstanz, July 2006

[22] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, P. Sommerlad: Security Patterns: Integrating Security and Systems Engineering, John Wiley & Sons, 2006

[23] D.C. Schmidt, M. Stal, H. Rohnert, F. Buschmann: Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects, John Wiley & Sons, 2000

[24] P. Jain, M. Kircher: Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management, John Wiley & Sons, 2004.

[25] F. Buschmann, K. Henney, D.C. Schmidt: Pattern-Oriented Software Architecture, Vol. 4

[26] Christopher Alexander, Sara Ishikawa, Murray Silverstein: A Pattern Language.

[27] P. Pelliccione, CHARMY: A framework for Software Architecture. PhD thesis, 2005.

[28] A. Bilgin, J. Ellson, E. Gansner, Y. Hu, and S. North, “Graphviz - Graph Visualization Software,” 2017.

[29] D. Compare, A. D’Onofrio, A. Di Marco, and P. Inverardi, “Automated performance validation of software design: an industrial experience,” *Proceedings. 19th International Conference on Automated Software Engineering*, 2004., pp. 298–301, 2004.

[30] A. Rademaker, C. Braga, and A. Sztajnberg, “A Rewriting Semantics for a Software Architecture Description Language,” *Electronic Notes in Theoretical Computer Science*, vol. 130, pp. 345–377, may 2005.

Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología Avanzada – RITA

REVISTA

TIA