NUEVAS REALIDADES PARA LA EDUCACIÓN EN INGENIERÍA: CURRÍCULO, TECNOLOGÍA, MEDIO AMBIENTE Y DESARROLLO
13 - 16 DE SEPTIEMBRE 2022 CARTAGENA DE INDIAS, COLOMBIA
Encuentro Internacional de Educación en Ingeniería ACOFI

Cloud testing y distributed testing como modelos de pruebas para sistemas de microservicios

Daniel Gómez Betancur, Christian Andrés Candela Uribe, Julio Cesar Chavarro Porras, Luis Eduardo Sepúlveda

Universidad Tecnológica de Pereira
Pereira, Colombia

John Alexander Sanabria

Universidad del Valle
Cali, Colombia

Resumen

Actualmente, la arquitectura de microservicios (MSA) ha ganado popularidad en la industria del software; especialmente al permitir la construcción de sistemas con componentes desacoplados que le aportan dinamismo y modularidad; sin embargo, esta arquitectura también presenta algunos desafíos, como el monitoreo y las pruebas. El objetivo de este trabajo es proponer un modelo de pruebas para MSA con la intención de mejorar la confiabilidad y estabilidad de las pruebas.

Este trabajo se desarrolló mediante dos etapas principales. En la primera etapa se identificaron modelos de prueba en Sistemas Distribuidos mediante búsquedas en bases de datos y búsquedas por bola de nieve. En la segunda etapa, utilizando los resultados del SMS, se realizó un proceso formal de selección para identificar el modelo más propenso a ser adaptado a MSA. Metodológicamente, este trabajo utilizó prácticas de la ingeniería de software basada en evidencias que fueron adoptadas para el SMS y el proceso de toma de decisiones.

En los resultados obtenidos, se identificaron los principales modelos de prueba en sistemas distribuidos; en base a esta información, se puntuaron los modelos identificados para determinar aquellos con mayor probabilidad de adaptación a sistemas de microservicios, priorizando la estabilidad y confiabilidad de la ejecución de las pruebas. Finalmente, se seleccionaron los modelos de Cloud

Testing y Distributed Testing como los más aptos para adaptarse a los sistemas de microservicios de acuerdo con el objetivo propuesto.

Palabras clave: MSA; sistemas de microservicios; pruebas de software

Abstract

Currently, microservice architecture (MSA) have gained popularity in the software industry, especially by allowing the building of software as decoupled components that bring dynamism and modularity to the system; however, this architecture also brings some challenges such as monitoring and testing. The objective of this work is to propose a testing model for MSA focusing on improving the reliability and stability of the tests.

This work has two main stages. First, to identify test models in distributed systems by performing database search and snowballing. Second, using the previous results, a formal process to select the most likely adaptable model to MSA. Methodologically, this work used some practices from evidence-based software engineering were adopted for SMS and decision-making processes.
Main test models in distributed systems were identified in the obtained results. Based on this information, the models obtained were scored to determine those most likely to be adapted to microservices systems, prioritizing the stability and reliability of test execution. Finally, Cloud Testing and Distributed Testing models were selected as the most likely to be adapted to microservices systems, highlighting the stability and reliability of the results.

Keywords*: MSA; microservices based systems; software testing*

1 Introducción

La arquitectura de microservicios (MSA) ha recibido atención en la academia y la industria debido a su compatibilidad con los entornos de computación en la nube (di Francesco et al., 2017). Si bien la implementación de MSA brinda beneficios como flexibilidad y escalabilidad (Hasselbring & Steinacker, 2017), MSA también enfrenta desafíos como el monitoreo y las pruebas (Alshuqayran et al., 2016; Vera-Rivera et al., 2019). Estos desafíos se ven agravados por el uso intensivo de la comunicación asíncrona que genera problemas relacionados con la consistencia de los datos (Engel et al., 2018).

Específicamente, se han evidenciado problemáticas como la alta complejidad, la baja confiabilidad, la falta de investigación y la falta de herramientas en la realización de pruebas (Candela Uribe et al., 2021). Este trabajo tiene como objetivo proponer una estrategia para la realización de pruebas en Sistemas de Microservicios, para la cual se identificaron los modelos de pruebas aplicados en Sistemas Distribuidos y se seleccionaron aquellos más propensos a ser adaptados a MSA, priorizando los que permitirían aumentar la confiabilidad de las pruebas y enriquecer el conjunto de herramientas e investigación. Finalmente, se decide incorporar en la estrategia

propuesta tecnologías emergentes como el Cloud Testing y Distributed Testing como alternativas para tener en cuenta en los procesos de pruebas.

Este documento está estructurado de la siguiente manera. La sección 2 presenta los trabajos relacionados. La Sección 3 describe la metodología para identificación y selección del modelo. La Sección 4 discute las amenazas a la validez. la Sección 6 presenta las conclusiones y la Sección 7 las referencias.

2 Trabajo Relacionado

En el trabajo "A Systematic Mapping Study on Microservices Architecture in DevOps", Waseem et al. (2020) identifican los inconvenientes en las pruebas de microservicios, las tendencias de investigación relacionadas con los atributos de calidad y las soluciones propuestas para los problemas identificados. Li et al. (2021) en "Understanding and addressing quality attributes of microservices architecture: A Systematic literature review", describen, analizan y clasifican la literatura relacionada con los sistemas de microservicios y DevOps. Candela Uribe et al. (2021), en su trabajo denominado "Problemas en la Implementación de Pruebas en Sistemas de Microservicios: Estudio de Mapeo Sistemático", identifican los principales problemas relacionados con las pruebas de microservicios y señalan las causas más relevantes de estos. Este trabajo complementa los presentados anteriormente, identificando los modelos de prueba más utilizados en sistemas distribuidos, y seleccionando aquellos adaptables a MSA.

3 Metodología

Dividimos este estudio en dos fases: un estudio de mapeo sistemático para identificar los principales modelos de prueba en sistemas distribuidos y un proceso de toma de decisiones para identificar el modelo más probable para adaptarse a MSA. Cada fase tiene sus etapas, ver Figura 1.

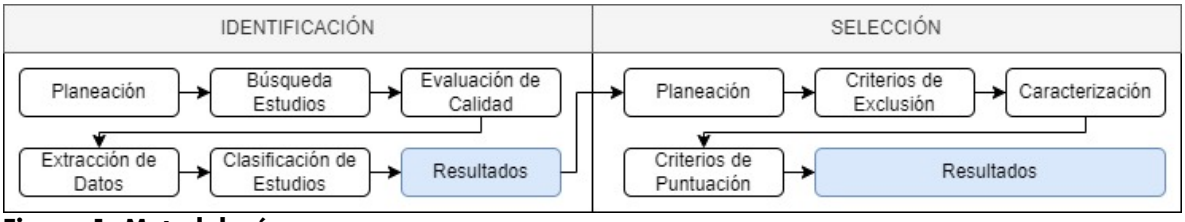

Figura 1. Metodología

3.1 Identificación

Basados en métodos ya implementados en la realización de Estudios de Mapeo Sistemático, se adaptó una estrategia híbrida combinando la búsqueda manual y automática de estudios (Mourao et al., 2017; Nguyen et al., 2015). Se siguió también la adaptación propuesta por Kitchenham et al. (2010), que se compone por las etapas: Planificación, Búsqueda de Estudios, Evaluación de la Calidad, Extracción de Datos, Clasificación de Estudios, Resultados.

3.1.1 Etapa 1: Planeación

En esta etapa se establecieron los Objetivos del Estudio, Preguntas de Investigación, Métricas, Tópicos de Clasificación, Criterios de Inclusión/Exclusión y Criterios de Calidad. Se aplicó el modelo GQM para los componentes Objetivos de estudio, Preguntas de investigación y Métricas (Basili, 1992; Basili & Caldiera, 2000). Ver Tabla 1.

Tabla 1. GQM

Elemento	Código	Descripción
Objetivo	G1	Determinar cuáles son los principales modelos de prueba de sistemas distribuidos durante el período 2010-2020.
Pregunta	P1	¿Cuáles son los modelos de prueba de sistemas distribuidos más estudiados?
	P2	¿Cómo se clasifican los principales modelos de Pruebas de Sistemas Distribuidos?
Métrica	M1	Cantidad de estudios relacionados con el modelo de pruebas de sistemas distribuidos

Los criterios de Inclusión y Exclusión se explican en la Tabla 2.

Tabla 2. Criterios de Inclusión y Exclusión

Categoría	Inclusión	Exclusión
Campos	Título, Resumen y Palabras Clave	-
Tipo de Publicación	Artículos de revista	Conferencias, Tesis, Libros
Área/Disciplina	Ingeniería de Software y Computación	Áreas no relacionadas a la computación
Periodo	2010 - 2020	Anterior a 2010
Idioma	Inglés	-

3.1.2 Etapa 2: Búsqueda de Estudios

Estrategia de Búsqueda

Se articularon dos estrategias de búsqueda. "Búsqueda en Base de Datos" y "Bola de Nieve"; el primero consiste en realizar una consulta mediante una cadena de búsqueda en las bibliotecas definidas para este estudio. El segundo, consiste en un proceso manual cuyo objetivo es encontrar más estudios a partir de las referencias y citaciones de los estudios más relevantes obtenidos en la primera estrategia. Las herramientas utilizadas para este SMS fueron las bases de datos, el software SMS Builder (Candela-Uribe et al., 2022), *Mendeley* y *Google Scholar*.

Búsqueda en Bases de Datos

Para identificar los estudios se utilizaron las bases de datos ACM, IEEE Xplore, Science Direct, Web of Science, Springer y Scopus. Se utilizó la metodología PICOC como guía para identificar términos y frases clave que ayuden a la búsqueda de estudios (Needleman, 2002; Petticrew & Roberts, 2008). Una vez se identificaron las palabras clave, se usó el operador booleano "OR" para agregar sinónimos a los términos de las cadenas de búsqueda. Para obtener una intersección de las diferentes palabras clave, se usó el operador booleano "Y" en las cadenas de búsqueda. Ver Tabla 3.

Tabla 3. Palabras Clave para las cadenas de búsqueda

Palabra Clave	Sinónimo
Testing Method	Testing schema, testing methodology, testing process, testing strategy, assessing method, assessing schema, assessing methodology, assessing process, assessing strategy, test method, test schema, test methodology, test process, test strategy, assess method, assess schema, assess methodology, assess process, assess strategy, assessment method, assessment schema, assessment methodology, assessment process, assessment strategy
Distributed System	Distributed architecture, service architecture, service-based system, microservices architecture, microservices based architecture, SOA, microservice system, microservices based system

Tras ejecutar las cadenas de búsqueda se identificaron 4741 estudios. Después de aplicar los criterios de Inclusión/Exclusión los resultados se redujeron a 443. Posteriormente, se eliminaron los artículos duplicados obteniendo un total de 407 estudios. Después de eso, inició la revisión de título, palabras clave y los resúmenes para eliminar aquellos que no están relacionados con los objetivos del SMS. Resultaron 89 artículos.

Búsqueda por Bola de Nieve

Como indicadores de calidad, usaron los índices CVI y SCI (Almanasreh et al., 2019). Éstos se aplicaron sobre los 89 artículos resultantes de la búsqueda en las bases de datos. Se incorporaron como línea de base 16 artículos con el CVI más alto y 18 artículos con el SCI más alto. En total, se seleccionaron 34 artículos diferentes como base de la bola de nieve. Se realizó una bola de nieve hacia atrás, en la que se identificaron 60 nuevos estudios para ingresar al SMS, luego se realizó la bola de nieve hacia adelante utilizando la herramienta Google Scholar propuesta por Ali et al. (2019), con esto, se encontraron 60 estudios. Después de este proceso, el número total de estudios identificados fue de 209.

Resultados de Búsquedas

Finalmente, se agregaron los estudios identificados por las dos estrategias. Por Búsqueda en bases de datos se agregaron 89 estudios mientras que por Bola de Nieve se agregaron 120. Las referencias de los estudios analizados se encuentran en el siguiente repositorio https://github.com/daniel33gomez/Testing-Models-Distributed-Systems

3.1.3 Etapa 3: Evaluación de Calidad

Se utilizaron los indicadores CVI, donde cada estudio se evalúa con una escala cuantitativa de 0 a 5 para identificar la relación con los objetivos de investigación (Almanasreh et al., 2019); el índice SCI (Study Citation Index), que representa el número de citaciones de los estudios; y el IRRQ para identificar la relación del estudio con respecto a las preguntas de investigación.

3.1.4 Etapa 4: Extracción de datos

Se identificaron 209 estudios primarios, de los que se extrajo la información referente a las preguntas de investigación propuestas en este estudio de mapeo sistemático.

3.1.5 Etapa 5: Clasificación de estudios

Los estudios se clasificaron de acuerdo con el tipo de modelo que presentaron: **Functional Testing, Non-Functional Testing** ó **Testing Execution**.

3.1.6 Etapa 6: SMS Results

En la Figura 2, se evidencia que los modelos de prueba más relevantes según la cantidad de estudios publicados son el Model-based Testing, Cloud Testing y Testing as a Service.

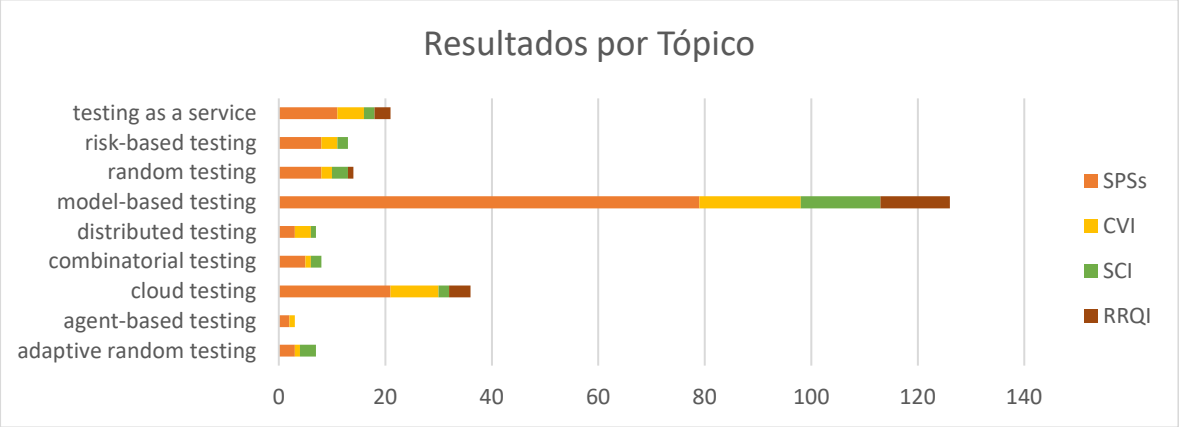

Figura 2. Resultados por Modelo de Pruebas

3.2 Selección

Para identificar la mejor opción de modelos que podrían adaptarse a MSA, se implementó un proceso de Análisis y Resolución de Decisiones (DAR por sus siglas en inglés), que permite analizar posibles decisiones durante el proyecto utilizando métodos de evaluación formal para elegir la mejor alternativa disponible. (Chaudhary & Chopra, 2017; O'Regan, 2017). El proceso se basa en establecer criterios de evaluación, a cada criterio definido se le otorgará una clasificación, y los criterios mejor clasificados tendrán más peso en el proceso de selección (Chaudhary & Chopra, 2017). Primero, se definieron los criterios de exclusión para eliminar los modelos que no se alinean a los objetivos de este trabajo. Como segundo paso, se implementaron los criterios de puntuación para identificar la mejor opción de modelo que se puedan adaptar a MSA.

3.2.1 Criterios de Exclusión

Los criterios de exclusión definidos se muestran en la Tabla 4. Para pasar a la siguiente etapa de toma de decisiones, los modelos deben cumplir con los criterios propuestos; de lo contrario, serán excluidos.

Tabla 4. Criterios de Exclusión

Criterio	Código	Tipo	Peso
Es un modelo de ejecución de pruebas	C01	Excludes	-
Ejecuta pruebas previas al despliegue en producción	C02	Excludes	-
Ejecuta pruebas funcionales	C03	Excludes	-

Los modelos que cumplen los criterios de exclusión mencionados son Cloud Testing & TaaS, Agent-based Testing y Distributed Testing. Estos modelos pasan a la segunda fase del proceso DAR.

3.2.2 Criterios de Puntuación

Los criterios de puntuación se basaron en los problemas y causas reportadas por Candela Uribe et al. (2021), teniendo en cuanta el nivel de afectación sobre los MSA. Ver Tabla 5.

Tabla 5. Criterios de Puntuación

Problemática reportada	Importancia	Criterio	Peso (w)
Alto consume de recursos	Baja	Consumo de recursos	1
Las pruebas son tediosas	Baja	Usabilidad del modelo	1
Dificultad para localizar el componente que falla	Baja	Herramientas de seguimiento	1
Las pruebas tienden a ser engañosas	Baja	Confiabilidad	1
Alta Complejidad	Alta	Complejidad	3
Tiempos de espera	Baja	Tiempos de espera	1
Falta de Herramientas	Media	Disponibilidad de Herramientas	2
Versiones de servicios	Media	Mantenimiento	2
Falta de Investigación	Alta	Recursos disponibles	3
Alto número de servicios	Alta	Escalabilidad	3

Se usó la escala de Likert para evitar el sesgo en la puntuación, cada valor en ésta mide el grado de relación del modelo hacia un elemento en particular (Liu et al., 2017). En base a esto, se utilizó una escala de Bueno, Regular y Malo, cuyos valores cuantitativos son 3 para Bueno, 2 para Regular y 1 para Malo. La siguiente ecuación representa el modelo de puntuación basado en la escala Likert.

$$s = \sum_{i=0}^{n} w_i \, P_i$$

Donde s es la puntuación final del modelo, w es el peso del criterio y P es la puntuación del modelo con respecto al criterio. Los resultados se pueden evidenciar en la Tabla 6.

Tabla 6. Puntuación de los Modelo de Prueba

Criterio	Cloud Testing	Distributed Testing	Agent-based Testing
Complejidad	Bueno	Malo	Malo
Consumo de recursos	Regular	Malo	Malo
Escalabilidad	Bueno	Bueno	Bueno
Confiabilidad	Bueno	Malo	Malo
Mantenimiento	Bueno	Malo	Malo
Usabilidad	Bueno	Malo	Malo
Herramientas de Seguimiento	Malo	Malo	Malo
Herramientas Disponibles	Bueno	Regular	Malo
Recursos disponibles	Bueno	Regular	Malo
Tiempo de espera	Regular	Malo	Malo
Final Score (s)	50	29	24

La información de cada criterio se extrajo de diferentes estudios. Ver Tabla 7.

Tabla 7. Recursos sobre caracterización de los modelos

Modelo	Referencias
Cloud Testing	(Gao et al., 2011; Shrivastva et al., 2014; Vijayaraghavan et al., 2020; Yao et al., 2019)
Distributed Testing	(Azzouzi et al., 2012, 2015; Hierons & Núñez, 2017; Mehmood et al., 2016)

Agent-based Testing	(Azzouzi et al., 2010, 2012, 2015; Ponnurangam & Uma, 2005)

3.3 Resultados

Los principales problemas presentes en las pruebas de sistemas de microservicios están relacionados con Complejidad, Consumo de recursos, Escalabilidad, Confiabilidad, Mantenimiento, Usabilidad, Trazabilidad y Disponibilidad de herramientas (Candela Uribe et al., 2021); De acuerdo con esta investigación, el modelo de Cloud Testing trae soluciones para enfrentar todos estos problemas, teniendo un servicio seguro, confiable y bajo demanda. Por su parte, el modelo de Pruebas Distribuidas podría aportar la escalabilidad y versatilidad necesarias a este tipo de sistemas.

4 Amenazas a la Validez

Se aplicó el protocolo de construcción de SMS sugerido en los estudios Budgen et al, (2008) y Kitchenham et al. (2010). La búsqueda en bases de datos se realizó en 6 bibliotecas de alto impacto en el contexto científico. La validación de los estudios se realizó a través de los índices CVI, SCI y RRQI. Para minimizar el sesgo en el proceso de tomas de decisiones, se utilizó una escala de Likert (Liu et al., 2017), la cual es popular en el análisis de información, donde cada valor de la escala representa el grado de relación entre los criterios definidos y el modelo analizado.

5 Conclusiones

Se identificaron los Modelos de Pruebas de Sistemas Distribuidos más relevantes de los estudios publicados entre 2010 y 2020, se identificaron el Model-based Testing, Cloud Testing y Distributed Testing como los modelos de prueba más relevantes. Posteriormente, se llevó a cabo un proceso de selección para determinar los modelos adaptables a MSA. Tras analizar las diferentes características de los modelos, este trabajo propone el uso de los modelos Cloud Testing y Distributed Testing como parte de una estrategia de pruebas con el fin de aportar confiabilidad y escalabilidad en la ejecución de pruebas en MSA.

6 Referencias

- Ali, A. Q., Sultan, A. B. M., Ghani, A. A. A., & Zulzalil, H. (2019). A Systematic Mapping Study on the Customization Solutions of Software as a Service Applications. *IEEE Access*, *7*, 88196–88217. https://doi.org/10.1109/ACCESS.2019.2925499
- Almanasreh, E., Moles, R., & Chen, T. F. (2019). Evaluation of methods used for estimating content validity. *Research in Social and Administrative Pharmacy*, *15*(2), 214–221. https://doi.org/10.1016/j.sapharm.2018.03.066
- Alshuqayran, N., Ali, N., & Evans, R. (2016). A systematic mapping study in microservice architecture. *Proceedings - 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications, SOCA 2016*, 44–51. https://doi.org/10.1109/SOCA.2016.15
- Azzouzi, S., Benattou, M., & Charaf, H. (2012). Test execution control with timing constraints for testing distributed systems. *Journal of Theoretical and Applied Information Technology*, *46*(1), 486–

498. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84874551626&partnerID=40&md5=e4a067422c93a1c5ffa40b81dd04a42b

- Azzouzi, S., Benattou, M., & Charaf, M. E. H. (2015). A temporal agent based approach for testing open distributed systems. *Computer Standards and Interfaces*, *40*, 23–33. https://doi.org/10.1016/j.csi.2015.01.003
- Azzouzi, S., Benattou, M., El, M., Charaf, H., & Abouchabaka, J. (2010). SMA and Mobile Agents Actors for Distributed Testing. *IJCSI International Journal of Computer Science Issues*, *7*(5), 231. www.IJCSI.org
- Basili, V. R. (1992). *Software Modeling and Measrement*.
- Basili, V. R., & Caldiera, G. (2000). The Goal Question Metric Paradigm. *Encyclopedia of Software Engineering - 2 Volume Set*, *528–532*. https://www.cs.umd.edu/~basili/publications/technical/T89.pdf
- Budgen, D., Turner, M., Brereton, P., & Kitchenham, B. (2008). Using Mapping Studies in Software Engineering. *Ppig*, *2*, 195–204. http://www.ppig.org/papers/20th-budgen.pdf
- Candela Uribe, C. A., Gómez-Betancur, D., Sepúlveda-Rodríguez, L., Chavarro-Porras, J., & Sanabria-Ordoñez, J. (2021). Problemas en la Implementación de Pruebas en Sistemas de Microservicios: Estudio de Mapeo Sistemático. *Investigación e Innovación En Ingenierías*, *9*(3 SE-Artículos), 3–17. https://doi.org/10.17081/invinno.9.3.5314
- Candela-Uribe, C. A., Sepúlveda-Rodríguez, L. E., Chavarro-Porras, J. C., Sanabria-Ordoñez, J. A., Garrido, J. L., Rodríguez-Domínguez, C., & Guerrero-Contreras, G. (2022). SMS-Builder: An adaptive software tool for building systematic mapping studies. *SoftwareX*, *17*, 100935. https://doi.org/https://doi.org/10.1016/j.softx.2021.100935
- Chaudhary, M., & Chopra, A. (2017). *CMMI Design BT - CMMI for Development : Implementation Guide* (M. Chaudhary & A. Chopra, Eds.; pp. 9–69). Apress. https://doi.org/10.1007/978-1-4842-2529-5_2
- di Francesco, P., Malavolta, I., & Lago, P. (2017). Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, 21–30. https://doi.org/10.1109/ICSA.2017.24
- Engel, T., Langermeier, M., Bauer, B., & Hofmann, A. (2018). Evaluation of microservice architectures: A metric and tool-based approach. In J. Mendling & H. Mouratidis (Eds.), *Lecture Notes in Business Information Processing* (Vol. 317, pp. 74–89). Springer International Publishing. https://doi.org/10.1007/978-3-319-92901-9_8
- Gao, J., Bai, X., & Tsai, W.-T. (2011). Cloud testing-issues, challenges, needs and practice. *Software Engineering: An International Journal*, *1*(1), 9–23.
- Hasselbring, W., & Steinacker, G. (2017). Microservice architectures for scalability, agility and reliability in e-commerce. *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, 243–246. https://doi.org/10.1109/ICSAW.2017.11
- Hierons, R. M., & Núñez, M. (2017). Implementation relations and probabilistic schedulers in the distributed test architecture. *Journal of Systems and Software*, *132*, 319–335. https://doi.org/10.1016/j.jss.2017.03.011
- Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O. P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering-A tertiary study. *Information and Software Technology*, *52*(8), 792–805. https://doi.org/10.1016/j.infsof.2010.03.006
- Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., & Babar, M. A. (2021). Understanding and addressing quality attributes of microservices architecture: A Systematic literature review. *Information and Software Technology*, *131*, 106449. https://doi.org/10.1016/j.infsof.2020.106449

- Liu, Q., Basu, D., Goel, S., Abdessalem, T., & Bressan, S. (2017). *How to Find the Best Rated Items on a Likert Scale and How Many Ratings Are Enough BT - Database and Expert Systems Applications* (D. Benslimane, E. Damiani, W. I. Grosky, A. Hameurlain, A. Sheth, & R. R. Wagner, Eds.; pp. 351–359). Springer International Publishing.
- Mehmood, M. A., Mahmood, A., Ahmed Khan, M. N., & Khatoon, S. (2016). A scenario-based distributed testing model for software applications. *International Journal of ADVANCED AND APPLIED SCIENCES*, *3*(10), 64–71. https://doi.org/10.21833/ijaas.2016.10.011
- Mourao, E., Kalinowski, M., Murta, L., Mendes, E., & Wohlin, C. (2017). Investigating the Use of a Hybrid Search Strategy for Systematic Reviews. *International Symposium on Empirical Software Engineering and Measurement*, *2017-Novem*, 193–198. https://doi.org/10.1109/ESEM.2017.30
- Needleman, I. G. (2002). A guide to systematic reviews. *Journal of Clinical Periodontology*, *29*(SUPPL. 3), 6–9. https://doi.org/10.1034/j.1600-051x.29.s3.15.x
- Nguyen, P. H., Kramer, M., Klein, J., & Traon, Y. le. (2015). An extensive systematic review on the Model-Driven Development of secure systems. *Information and Software Technology*, *68*, 62–81. https://doi.org/10.1016/j.infsof.2015.08.006
- O'Regan, G. (2017). *Capability Maturity Model Integration BT - Concise Guide to Software Engineering: From Fundamentals to Application Methods* (G. O'Regan, Ed.; pp. 255–277). Springer International Publishing. https://doi.org/10.1007/978-3-319-57750-0_16
- Petticrew, M., & Roberts, H. (2008). Systematic Reviews in the Social Sciences: A Practical Guide. In *Systematic Reviews in the Social Sciences: A Practical Guide*. John Wiley & Sons. https://doi.org/10.1002/9780470754887
- Ponnurangam, D., & Uma, G. v. (2005). Fuzzy complexity assessment model for resource negotiation and allocation in agent-based software testing framework. *Expert Systems with Applications*, *29*(1), 105–119. https://doi.org/10.1016/j.eswa.2005.01.008
- Shrivastva, A., Gupta, S., & Tiwari, R. (2014). Cloud based Testing Techniques (CTT). *International Journal of Computer Applications*, *104*(5), 24–29. https://doi.org/10.5120/18200-9122
- Vera-Rivera, F. H., Gaona Cuevas, C. M., & Astudillo, H. (2019). Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación. *Revista Ibérica de Sistemas e Tecnologias de Informação*, *E23*(E23), 107–120.
- Vijayaraghavan, V., Singh, A. R., & Sucharita, S. (2020). *A Modern Perspective on Cloud Testing Ecosystems* (M. Ramachandran & Z. Mahmood, Eds.; pp. 255–276). Springer International Publishing. https://doi.org/10.1007/978-3-030-33624-0_10
- Waseem, M., Liang, P., & Shahin, M. (2020). A Systematic Mapping Study on Microservices Architecture in DevOps. *Journal of Systems and Software*, *170*, 110798. https://doi.org/10.1016/j.jss.2020.110798
- Yao, J., Maleki Shoja, B., & Tabrizi, N. (2019). An overview of cloud computing testing research. In D. da Silva, Q. Wang, & L.-J. Zhang (Eds.), *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11513 LNCS* (pp. 303–313). Springer International Publishing. https://doi.org/10.1007/978-3-030-23502-4_21

Sobre los Autores

- MSc(c) **Daniel Gómez-Betancur**, Estudiante de Maestría en Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira, daniel33.gomez@utp.edu.co
- PhD(c) **Christian Andrés Candela-Uribe**, Estudiante Doctorado en Ingeniería, Universidad Tecnológica de Pereira-Universidad del Quindío, christiancandela@uniquindio.edu.co

- PhD(c) **Luis Eduardo Sepúlveda**, Estudiante Doctorado en Ingeniería, Universidad Tecnológica de Pereira - Universidad del Quindío, lesepulveda@uniquindio.edu.co
- PhD **Julio César Chavarro-Porras**, Profesor, Universidad Tecnológica de Pereira, jchavar@utp.edu.co
- PhD **John Alexander Sanabria Ordóñez**, Profesor, Universidad del Valle, john.sanabria@correounivalle.edu.co

---